uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería en Tecnologías de Telecomunicación
2017-2018

Trabajo Fin de Grado

"Configuración de un Entorno de Laboratorio de VoIP para Docencia"

José Antonio Espinosa Melchor

Tutor

Iván Vidal Fernández

Leganés, 2018

RESUMEN

El uso cada vez más extendido de Internet, como plataforma global de comunicación, engloba desde los mensajes más sencillos a videollamadas entre múltiples usuarios. En este marco está comprendida la comunicación de VoIP (*Voice over IP*, Voz sobre IP) y, por tanto, se hace necesaria la inclusión de su estudio en el ámbito de la telemática.

En la última década los servicios de VoIP se han convertido en la opción preferida a la hora de actualizar o instalar un nuevo sistema telefónico. Es una forma económica y sencilla que concede a quien establece este tipo de servicio un control total, sin necesidad de recurrir a una infraestructura que requiera de una persona con mucha preparación para su instalación, modificación o mantenimiento. Además, permite la integración de diversos *software* y dispositivos ya existentes, o la conexión con otros centros sin necesidad de que haya distintas PBX (*Private Branch Exchange*, Ramal Privado de Conmutación).

En este escrito se presenta una posible solución para su aplicación en unas prácticas en nuestro contexto educativo, en el que se podrán generar y terminar llamadas y conferencias entre múltiples terminales, ya sean móviles *Wi-Fi*, *software* o aplicaciones.

Así pues, el trabajo está consiste en diseñar y desarrollar un sistema de comunicación VoIP orientado a la mejora de unas prácticas ya existentes, en las que se aplicarán DNS (*Domain Name System*, Sistema de Nombre de Dominio). En relación con el servicio de transmisión de voz se utilizarán dos programas, Kamailio y Asterisk, de forma integrada, de manera que Kamailio actuará como *proxy*; y Asterisk será el encargado de la gestión de las llamadas entre los terminales.

Palabras clave: SIP (*Session Initiation Protocol*); VoIP (*Voice over IP*); DNS (*Domain Name System*); *Proxy*

ABSTRACT

The increasingly use of the Internet, as a global platform for communications, encompass from simple messages to videocalls between multiple users. It is in this sphere where VoIP (Voice over IP) is included, and therefore it's study is required in the scope of telematics.

In the last decade VoIP services have become the preferred choice when updating or installing a new telephony system. It is a simple and economical way to grant the owner total control, no need of an infrastructure nor a high-qualified person to take charge of installation, modification or maintenance. It also permits integration of diverse and already existing software and devices, or connection with other centers without having to get through different PBX (Private Branch Exchange).

In this thesis a possible solution to its application in the context of university practices is presented, where calls or conferences can be generated and terminated between multiple devices, either wi-fi phones, software or applications.

Consequently, this study consists on designing and developing a VoIP communication system intended to improve already existing practices by using DNS (Domain Name System). Voice transmission will be administrated by two programs integrated, Kamailio and Asterisk, where Kamailio acts as proxy; and Asterisk manages calls.

Key words: SIP (Session Initiation Protocol); VoIP (Voice over IP); DNS (Domain Name System); Proxy

MARCO REGULATORIO

Respetando las pautas de la ley orgánica 15/1999, de 13 de diciembre de Protección de datos de carácter personal, (LOPD) [LOPD, 1999] la plataforma de comunicación multimedia desarrollada para este proyecto no almacenará información de carácter personal de los usuarios, el contenido compartido por los usuarios será eliminado tras la terminación de la sesión.

El sistema cumple las regulaciones de interés público descritas en la Directiva 2010/13/UE del Parlamento Europeo y del Consejo, de 10 de marzo de 2010, sobre la coordinación de determinadas disposiciones legales, reglamentarias y administrativas de los Estados miembros relativas a la prestación de servicios de comunicación audiovisual (Directiva de servicios de comunicación audiovisual).

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este primer capítulo se describe la situación actual de las comunicaciones VoIP y se explica el porqué de la realización de este trabajo. Así mismo, se introducen los objetivos a cumplir con la realización del mismo y se describe la estructura que sigue la memoria.

1.1. Motivación del trabajo

La realización de este trabajo nace de la necesidad de familiarizarse con el creciente uso de la comunicación por voz vía IP (*Internet Protocol*, Protocolo de Internet). Según la CNMC (Comisión Nacional de los Mercados y la Competencia) en el año 2007 existían en España 375.000 líneas de VoIP. En el año 2016 sólo Movistar cerró el año con 2,9 millones de líneas VoIP. Y poco a poco aparecen en el mercado más servicios que permiten este tipo de prácticas. Algunos de estos servicios, como Skype o las llamadas de WhatsApp, los podemos encontrar a diario, por lo que entender su funcionamiento resulta fundamental en el área de las telecomunicaciones.

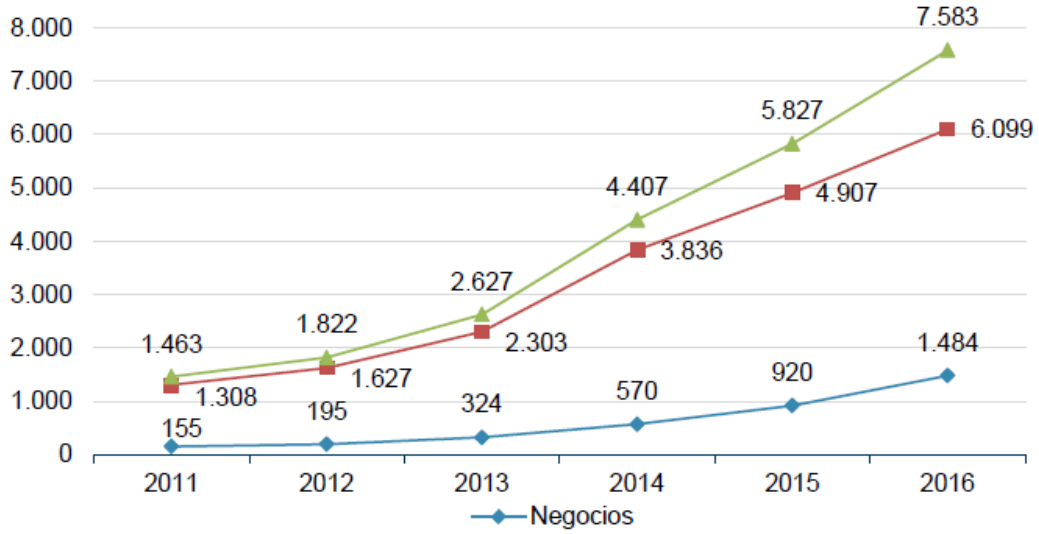

Fig. 1 Líneas de telefonía fija sobre Voz IP (miles de líneas) [1]

Por parte de las compañías telefónicas existen, sin embargo, algunas limitaciones en cuanto a su uso, ya que este tráfico no se transmite mediante la RTC (Red Telefónica Conmutada) y esta podría caer en desuso debido a las ventajas que VoIP ofrece. No solamente se envían paquetes de voz, sino que en la negociación de la sesión se puede incluir vídeo y la infraestructura a desplegar que es necesaria es relativamente simple, ya que se transmite por Internet y se pude tomar como base la que ya está disponible, por lo que tan solo precisa de una conexión estable, servidores *proxy* y registradores, lo que permite que a su vez sea más escalable.

A pesar de la existencia de restricciones por parte de las compañías, para su utilización en un entorno cerrado como puede ser una empresa, se pueden aprovechar al máximo esas ventajas, de manera que se pueden personalizar y gestionar llamadas, conferencias, permisos, etc.

Dada la importancia de los servicios VoIP estos forman parte del plan de estudios de la carrera de Ingeniería de Tecnologías de Telecomunicación. Es un tema estudiado, más concretamente, en la asignatura optativa de cuarto curso de Servicios Audiovisuales, cuyas prácticas docentes comprenden a nivel general este tema, por lo que se busca la mejora de dichas prácticas para aplicarse en el curso académico 2018/19.

1.2. Objetivo

Los objetivos principales de este trabajo son, por un lado, la implementación de un entorno de laboratorio de VoIP que pueda ser utilizado en prácticas docentes, de acuerdo con los estándares establecidos en los RFCs (*Request For Comments*) correspondientes a los protocolos utilizados en cada nivel de la comunicación, así como ahondar en los conocimientos previos acerca de los sistemas VoIP. Estos objetivos principales, a su vez, comprenden los siguientes subobjetivos:

- Despliegue de un servicio DNS (*Domain Name System*, Sistema de Nombre de Dominio) para soportar la identificación de usuarios y servidores de VoIP mediante URIs.

- Soporte de múltiples terminales, tanto *hardware* como *software*. Para ello la universidad cuenta con móviles VoIP *Wi-Fi* modelo ZyXEL p-2000w_v2 que permite la asignación de direcciones IP, números de teléfono o nombres de usuario y contraseña para identificarse, que implementan además los protocolos necesarios, como SIP o RTP. Además, existen diversas aplicaciones para PC o *smartphone* que permiten emular este mismo comportamiento.

- Creación de conferencias para soportar el establecimiento de llamadas entre múltiples terminales de usuario.

- El entorno se fundamentará en protocolos y estándares de Internet. Concretamente SIP (*Session Initiation Protocol*, Protocolo de Inicio de Sesión) un protocolo de señalización que permite crear, modificar y terminar sesiones entre varios participantes; SDP (*Session Description Protocol*, Protocolo de Descripción de la Sesión) utilizado para acordar una serie de parámetros que posibilite la comunicación entre ellos; RTP (*Real Time Protocol*, Protocolo en Tiempo Real) para transmitir audio y vídeo en tiempo real entre los usuarios.

- Se pretende implementar un sistema de bajo coste, por lo que su desarrollo se realizará mediante herramientas y soluciones de código abierto.

- Validación del entorno desarrollado mediante una batería de pruebas adecuadas que se ejecutarán en los laboratorios del Departamento de Ingeniería Telemática.

1.3. Estructura de la memoria

El contenido de este documento se encuentra dividido en distintos apartados agrupados por temas, y explicados a continuación:

- Capítulo 1: Introducción
  Explicación de la situación actual en la que se encuentran las comunicaciones VoIP, se presentan los motivos para realizar el proyecto y se establecen los objetivos a cumplir para la realización de este.

- Capítulo 2: Estado del arte
  Descripción de los principales protocolos y tecnologías utilizadas.

- Capítulo 3: Implementación
  En este apartado se detalla cómo está estructurado el sistema, así como la configuración de cada uno de los elementos que lo forman.

- Capítulo 4: Pruebas
  Se presentan una serie de exámenes realizados y las conclusiones obtenidas con el fin de comprobar que el sistema establecido funciona correctamente.

- Capítulo 5: Planificación y presupuestos
  Se describe el proceso de elaboración del proyecto, así como el coste asociado al mismo.

- Capítulo 6: Conclusiones
  Se describen las conclusiones obtenidas tras el desarrollo del proyecto y si se cumplen los propósitos descritos. Además, se exponen una serie de mejoras y acciones que se pueden realizar en el futuro partiendo de lo que ya se ha elaborado.

- Capítulo 7: Bibliografía
  Listado de referencias y fuentes de información utilizadas para la elaboración del proyecto

- Capítulo 8: Anexos
  Contienen información adicional acerca de alguno de los apartados anteriores. Hay cinco anexos:
  - Anexo A: Formato mensajes DNS
  - Anexo B: Descriptores SDP
  - Anexo C: Formato RTP y RTCP

2. ESTADO DEL ARTE

En este capítulo se describen los protocolos que serán utilizados a lo largo del proyecto y la razón por la que son necesarios.

2.1. DNS

El Sistema de Nombres de Dominio (DNS) [3] es un sistema que permite asignar a un nombre, como puede ser la dirección de una página web, un número que corresponde con su IP. Fue definido por primera vez en 1983 en los rfc882 y rfc883.

Las direcciones IPv4 consisten en números de 32 bits que no son fácilmente recordables, y en el caso de IPv6 se trata de 128 bits, por lo que es necesaria una manera de "traducir" esos valores a un lenguaje que se pueda reconocer y entender. Así por ejemplo la dirección 176.58.10.138 corresponde al dominio "uc3m.es".

Escribiendo "uc3m.es" en el buscador se envía una consulta a la red para buscar la IP asociada. Esta consulta la recibirá primero el servidor raíz, que localizará el servidor TLD (*Top Level Domain*, Dominio de Nivel Superior) más adecuado para seguir con la búsqueda. En nuestro caso el dominio de nivel superior será ".es". El siguiente paso lo dará el servidor TLD, que hará algo parecido, esta vez con el nombre de dominio (uc3m). El *host* devolverá la dirección IP que finalmente será la que utilice el usuario para acceder al sitio web. Este proceso lo realizaremos a nivel local utilizando BIND.

2.1.1. Resource Records

Registros de Recursos (RRs) [4] son los tipos de datos de DNS. En la respuesta, la información que se envía son estos RRs. El formato que siguen es:

<*domain name*> <TTL> IN <*record type*> <*value*>

- *Domain Name*: el nombre del dominio al que se hace referencia.
- *Time To Live* (TTL): el tiempo que durará almacenado en cache.
- IN: identifica al RR como un recurso de Internet.
- *Record Type*: tipo de recurso.
- *Value*: información específica de cada recurso.

TABLA 1. TIPOS DE RESOURCE RECORDS

Tipo	Descripción	Función
A	Dirección IPv4	Devuelve la dirección IP de 32 bits correspondiente al host que se indica. Puede haber varias IPs por cada *host*, o varios *hosts* por cada IP.
AAAA	Dirección IPv6	Igual que A pero con una dirección de 128 bits.

ANY	Todos los recursos en *cache*	Devuelve todos los recursos de todos los tipos.
CNAME	*Canonical Name* (Nombre Canónico)	Alias de un nombre de dominio.
MX	*Mail Exchange* (Intercambio de Correo)	Apuntan a un *host* que puede recibir mensajes como si fuera el principal.
NS	*Name Server* (Nombre del Servidor)	Identifica a los servidores y permite reconocerse entre sí.
PTR	*Pointer* (Puntero)	Es el opuesto a la A, asocia la dirección dada a un *host*. Apunta a un CNAME.
SIG	*Signature* (Firma)	Firma.
SOA	*Start Of Authority* (Inicio De Autoridad)	Indica el inicio de una nueva zona, información relevante y cuál es la fuente de esa información dentro del dominio. Debe haber solamente un SOA para cada subdominio.
SRV	*Service Locator* (Localizador de Servicio)	Utilizado para localizar algunos servicios como SIP, donde se definen aspectos como prioridad, peso, puerto, etc.
TXT	*Text* (Texto)	Datos extra como cifrado o claves de dominio.
Fuente: RFC 882		

Los que utilizaremos principalmente son: SOA, A, NS, PTR y SRV.

2.1.2. Formato

Todos los mensajes están divididos en cinco partes:

HEADER
QUESTION
ANSWER
AUTHORITY
ADDITIONAL

El valor de los campos *HEADER* y *QUESTION* lo determina el que realiza la petición y contienen un identificador de esa petición, otro para el tipo de solicitud, la clase de operación que se debe realizar y otros complementos específicos que ayudaran a dar una respuesta clara y completa. Los campos *ANSWER*, *AUTHORITY* Y *ADDITIONAL* pueden estar vacíos en ciertos casos. La descripción de cada uno de estos elementos se detalla en el Anexo A.

2.2. SIP

Session Initiation Protocol, o Protocolo de Inicio de Sesión en español [6][7], es un protocolo de señalización a nivel de aplicación creado a finales de los 90 para establecer, modificar y terminar sesiones con uno o más participantes que pueden incluir llamadas telefónicas a través de Internet, distribución de contenido multimedia y conferencias.

SIP utiliza elementos llamados servidores *proxy* que se encargan de tareas como enviar las solicitudes a los usuarios, autenticarlos, autorizarlos o registrarlos. Sigue una estructura similar a la utilizada por HTTP, con mensajes de texto, que además puede usarse sobre varios protocolos de transporte, como TCP y UDP.

Para permitir la comunicación entre los usuarios, SIP se complementa con otros protocolos como SDP, para acordar una serie de parámetros; y RTP para la transmisión de la información. El encaminamiento de SIP es independiente de las demás sesiones, como RTP, la negociación puede realizarse a través del *proxy* y la comunicación ser directa entre los usuarios.

2.2.1. Formato

Los usuarios se identifican mediante una URI (*Uniform Resource Identifier*, Identificador Uniforme de Recurso), que sigue un formato similar al de una dirección de correo electrónico:

sip:<username>@<domain.com>;<parameters>

Ej: host1@sip.uc3m.es

- *Username*: nombre de usuario al que identifica. Va seguido de una @.
- *Domain*: nombre del dominio o dirección IP. Puede contener un número de puerto precedido de ":".
- *Parameters*: opcionalmente se pueden especificar características como el protocolo de transporte a utilizar. Van separados por ";".

2.2.2. User Agents

Los UAs (*User Agent*, Agente de Usuario) son entidades lógicas que pueden actuar tanto como clientes (UAC), como servidor (UAS), dependiendo del papel que tengan. Son los que implementan el protocolo SIP: teléfonos móviles, ordenadores, etc.

- UAC

  Crea una petición y la envía. Solamente asume el estatus de UAC durante el tiempo que dura esa petición, si más adelante recibe una petición pasa a actuar como un UAS.

- UAS

  Al contrario que el UAC genera una respuesta a una petición SIP. Esta respuesta puede ser de aceptación, rechazo o redirección. El *software* que actúa como UAS lo hace únicamente mientras dura la transacción.

- B2BUA

  B2BUA (*Back-to-Back User Agent*, Agente de Usuario Espalda contra Espalda). Es una entidad que recibe y procesa peticiones como si fuera un UAS, pero actúa también como UAC para determinar cómo debe ser la respuesta, pudiendo enviar otras peticiones. Mantiene el estado y debe participar en todas las peticiones de los diálogos que ha iniciado. Puede desconectar y reconectar usuarios de una sesión, terminarlas, modificarlas, etc. Se utilizan, por tanto, para implementar servicios.

2.2.3. Servidores

Un servidor es un elemento que recibe peticiones y envía respuestas. Pueden ser *proxies*, UASs, servidores de redirección o *registrar*.

2.2.3.1. Proxy

Actúa como servidor y cliente. Su objetivo principal es asegurar que las peticiones o respuestas que recibe se acerquen al destinatario. El camino que siguen las respuestas es el opuesto al de las peticiones, pudiendo pasar por varios servidores *proxy*. Tiene autoridad para tomar decisiones y reescribir ciertas partes del mensaje que recibe si es necesario, aunque solamente trata la cabecera. Además, tiene dos modos de operación, *stateless* y *stateful*.

- *Stateless*: si no mantiene el estado, es decir, simplemente reenvía lo que recibe a su siguiente destinatario. Una vez enviado elimina toda la información del mensaje.
- *Stateful*: si mantiene el estado, almacena esa información y la utiliza para procesar otros mensajes que puedan llegar relacionados, pudiendo enviar una solicitud a varios destinatarios.

En el siguiente esquema se observa cómo uno de los terminales (UA) envía una solicitud INVITE al otro a través del proxy utilizando la URI SIP que identifica al receptor, en este caso el terminal de Alice llama a Bob usando “bob@dominio.com”. A continuación, el servidor reenvía esta petición a su destinatario que recibe la notificación y empieza a sonar hasta que da una respuesta, que será transmitida al servidor y de ahí al iniciador de la llamada. Finalmente se establece la comunicación por voz mediante el protocolo RTP.

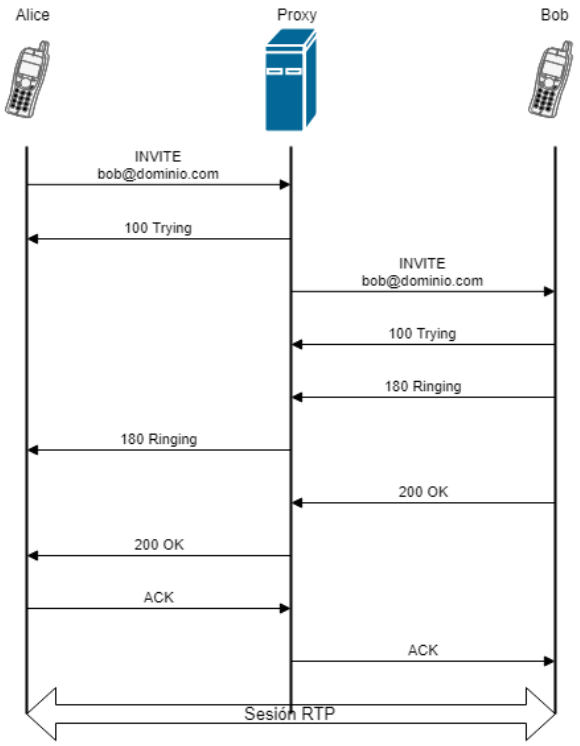

Fig. 2 Diagrama de un establecimiento de llamada

2.2.3.2. Redirect

Son servidores que remiten una respuesta *3xx*, lo que significa que informan al solicitante de que la petición que ha solicitado no puede procesarse, y presenta URIs alternativas que el cliente puede utilizar para satisfacer la petición. Un servidor *proxy* puede actuar también como *redirect*.

2.2.3.3. Registrar

Es un tipo de servidor que procesa peticiones del tipo REGISTER, que sirven como inscripción de cada usuario dentro del dominio, y se encarga de su localización. El *registrar* debe ser capaz de leer y escribir la información en un servicio de localización, y un servidor *proxy* o *redirect* debe ser capaz de leer esta misma información. También puede actuar como *proxy*.

Cuando recibe una petición REGISTER:

- Comprueba si el solicitante está en el dominio adecuado, reenviándola al servidor correspondiente si no es así.
- Comprueba que las cabeceras son correctas y autentica al UAC
- Determina si el usuario está autorizado para registrarse.
- Obtiene las distintas cabeceras la dirección a registrar, dirección de contacto o el tiempo durante el que será válido.
- Responde con mensaje “200 (OK)” si se ha registrado con éxito.

2.2.4. Mensajes

SIP es un protocolo basado en texto similar a HTTP, por lo que resulta legible. Los mensajes están formados por una línea inicial, una o más cabeceras y un cuerpo del mensaje opcional. Existen dos tipos de mensajes: solicitudes y respuestas. La primera línea, cada cabecera y la línea vacía tienen que terminar en un CLRF (*Carriage-Return Line-Feed*, Retorno de Carro y Salto de Línea).

<línea inicial>
<cabeceras del mensaje>
CLRF
<cuerpo del mensaje>

A continuación se muestra un ejemplo de mensaje SIP, específicamente una solicitud INVITE usada para establecer una llamada.

Fig. 3 Ejemplo de mensaje SIP

Como se puede observar, se indica tanto el protocolo utilizado (SIP) como el tipo de mensaje que es (INVITE). El mensaje consta de unas cabeceras que muestran información como el origen, el destino o por dónde ha pasado. Incluye además un cuerpo de mensaje en el que el protocolo SDP ofrece unas condiciones a cumplir para establecer la comunicación.

2.2.4.1. Solicitudes

La primera línea es la de solicitud. Contiene un nombre de método, una URI y la versión que se utiliza separados por un espacio (SP).

<método> SP <URI-SIP> SP <versión>

- Método: determinan el tipo de solicitud. Hay seis tipos básicos presentes en el primer estándar rfc3261:
  - *REGISTER*: para registrar información de contacto.
  - *INVITE*: inicia una sesión multimedia desde un UA.

    - *ACK*: respuesta definitiva a una solicitud INVITE.
    - *CANCEL*: cancela una solicitud INVITE que todavía esté pendiente.
    - *BYE*: termina la sesión.
    - *OPTIONS*: pide información sobre las competencias.
- URI-SIP: indica la dirección del usuario o servicio al que se envía la solicitud.
- Versión: la versión de SIP. Será siempre "SIP/2.0".

Las sesiones pueden ser modificadas enviando una nueva petición INVITE (RE-INVITE) a la que no hace falta responder con un "180 Ringing".

El número de métodos ha sido ampliado en RFCs más recientes: PRACK (3262), SUBSCRIBE, NOTIFY (3265), UPDATE (3311), REFER (3515), MESSAGE (3428), INFO (2976).

2.2.4.2. Respuestas

En este caso la primera línea indica el estado.

<versión> SP <código de estado> SP <frase>

- Versión: es la misma que en la solicitud "SIP/2.0"
- Código de estado: un código de tres dígitos que informa de la situación de la solicitud
- Frase: una breve explicación acerca de la respuesta dada.

El primer dígito del código define el tipo de respuesta, y los otros dos especifican la condición.

TABLA 2. CÓDIGOS DE RESPUESTAS SIP

Código	Tipo	Significado
1xx	Provisional	Se está procesando
2xx	Éxito	Se acepta la petición
3xx	Redirección	Se tiene que realizar otra acción para completar la petición
4xx	Error en el cliente	Error en la sintaxis o el servidor no puede cumplir la petición
5xx	Error en el servidor	El servidor no puede responder a una petición aparentemente válida

6xx	Fallo global	No se puede responder desde ningún servidor
Fuente: RFC 3261		

2.2.4.3. Cabeceras

Siguen una estructura:

<nombre cabecera>:<valores>,<más valores>

- Nombre cabecera:
  - *Via*: contiene las entidades que se han atravesado y cada *proxy* añade la suya propia. Indica la localización donde debe mandarse la respuesta y es recorrido en orden inverso.
  - *From*: identifica el inicializador lógico de la sesión. Contiene una URI. Está marcado con un *tag*, creado por el emisor y establecido en el *From* del primer mensaje.
  - *To*: identifica el receptor lógico. Contiene una URI. Al igual que el campo *From*, tiene un *tag* que se obtiene del *From* del primer mensaje recibido del *To* al que hace referencia.
  - *CSeq*: número de secuencia que indica el orden de las transacciones.
  - *Call-ID*: identificador único de la sesión.
  - *Max-Forwards*: establece un máximo de saltos que puede dar una solicitud. Sirve para evitar bucles de encaminamiento.
  - *Contact*: proporciona las URI-SIPs que se pueden usar para enviar solicitudes al UA
- Valores: información que recoge. Puede tomar varios si se separan por “,”.

No se distingue entre mayúsculas y minúsculas, a no ser que el contenido esté entre comillas.

2.2.5. SDP

Protocolo de Descripción de la Sesión, del inglés *Session Description Protocol*, [11] es un protocolo utilizado para describir sesiones multimedia, es decir, comunicar entre los participantes de la sesión la información de medios necesaria para poder participar. Se incluye en el cuerpo del mensaje SIP. Permite crear sesiones públicas o privadas, estando estas últimas cifradas. En el primer mensaje SIP, cuando se establece la llamada, el creador propone una lista de condiciones que puede cumplir para que la comunicación sea efectiva, por ejemplo, el tipo de códecs de audio que soporta. De esta manera, cuando el otro participante lo recibe, puede comparar esta lista con la suya propia y elegir en qué circunstancias se llevará a cabo.

SDP incluye información acerca del nombre de la sesión y propósito, tiempo que lleva activa, datos multimedia usados o cómo deben ser recibidos (dirección, puerto, formato, etc.), así como el tipo, protocolo de transporte y formato referentes a esos datos. Puede incluir información adicional, tal como el ancho de banda y datos de contacto del creador.

2.2.5.1. Formato

Consiste en una serie de líneas con el formato:

<type>=<value>

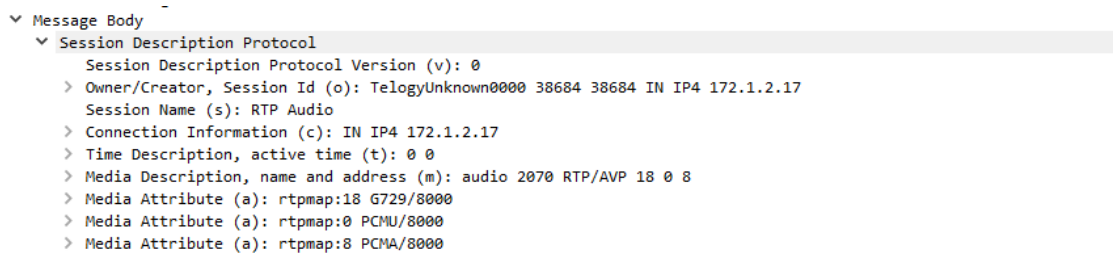

Fig. 4 Ejemplo de SDP

Esta información forma parte del cuerpo del mensaje INVITE (Fig. 3) y contiene los principales parámetros para la comunicación. La descripción de la sesión empieza con una línea “v=” que sirve además para terminar una posible descripción anterior. El orden que siguen las líneas es fijo, aunque algunas pueden omitirse.

- *Type*: siempre está formado por un solo carácter que actúa como identificador.
    - Descripción de la sesión:
        - v: versión del protocolo
        - o: identificador del creador
        - s: nombre de la sesión
        - i: información de la sesión (opcional)
        - u: URI de la descripción (opcional)
        - e: correo electrónico (opcional)
        - p: número de teléfono (opcional)
        - c: información de conexión (opcional)
        - b: información de ancho de banda (opcional)
        - z: ajuste de zona horaria (opcional)
        - k: clave de cifrado (opcional)
        - a: atributos de sesión (opcional)
        - (cero o más descriptores multimedia)
    - Descripción de tiempo:
        - t: tiempo que lleva activa la sesión
        - r: tiempo de repetición (opcional)
    - Descripción multimedia:
        - m: nombre del medio y dirección de transporte
        - i: título del medio (opcional)

        - c: información de conexión (opcional)
        - b: información de ancho de banda (opcional)
        - k: clave de cifrado (opcional)
        - a: líneas de atributo (opcionales)
- *Value*: es una cadena de caracteres que depende del *type*

La especificación de la sesión consiste en un nivel de descripción de sesión, que se aplica a toda la sesión y tráfico multimedia; y otro multimedia, que se aplica solamente a un tráfico concreto.

Los valores que pueden tomar cada descriptor se explican en el Anexo B, aunque ya se pueden observar ciertas características, por ejemplo, que el número de la versión es 0, y lo será en todos los casos; o que pueden incluirse varios descriptores multimedia, cada uno con unas propiedades que, según la situación, ofrecerán ciertas ventajas e inconvenientes.

2.2.6. Conferencias

El protocolo SIP permite que existan sesiones multimedia entre dos usuarios. Sin embargo, estas mismas sesiones entre múltiples usuarios son más complicadas y puede realizarse de varias maneras, sin que haya señalización y usando *multicast*; con señalización y uso de SIP siendo los participantes quienes realizan el control; o usando SIP y añadiendo un punto central en el que se hace el control de los usuarios. Se aplicará este último caso.

La arquitectura se basa en un punto central, llamado foco, que mantiene la señalización y las notificaciones SIP con cada uno de los participantes y es responsable de que los flujos de comunicación lleguen a todos ellos mediante el uso de uno o más *mixers*. Cada conferencia tiene un único foco y está identificada por una URI única que identifica a ese foco. Esto permite que los usuarios accedan a las conferencias enviando una solicitud INVITE a la URI específica y, de igual manera, deberán enviar una solicitud BYE para abandonarla.

Los mezcladores se encargan de combinar los flujos de datos de la conferencia y generar una o más salidas que distribuyen a los receptores, ya sean los participantes u otros *mixers*. Recibe las directrices de cómo hacerlo del foco, ya que los propios *mixers* no son "conscientes" por sí mismos de la existencia de la conferencia.

La creación de las conferencias supone la creación de un foco y unas políticas propias, y puede hacerse enviando una solicitud INVITE a una aplicación que la cree automáticamente, o que directamente un foco que se encuentre en un extremo la establezca y distribuya a otros extremos. Todo esto puede ejecutarse en un solo servidor que englobe tanto el foco como los *mixers* y en el que estén definidas las políticas para la interacción entre ellos, creación de conferencias, adición de usuarios, etc.

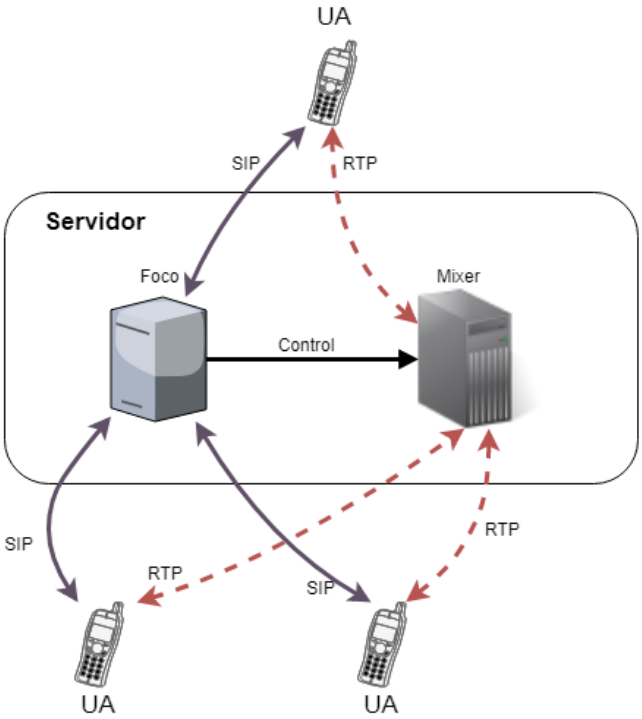

Fig. 5 Arquitectura de un servidor de conferencias SIP

2.3. RTP

Real-time Transport Protocol (Protocolo de Transporte en Tiempo Real) [12] es un protocolo que permite el intercambio de datos, como audio o vídeo, en tiempo real entre aplicaciones extremo a extremo. La monitorización de los datos transportados se hace utilizando el protocolo RTCP (*RTP Control Protocol*, Protocolo de Control RTP).

Será utilizado de manera complementaria a SIP para conseguir la comunicación entre usuarios, una vez hayan acordado los parámetros necesarios para ello.

RTP opera típicamente sobre UDP. Por sí mismo no reserva recursos, ni garantiza QoS (*Quality of Service*, Calidad de Servicio) o entrega ordenada de paquetes, sino que depende de capas inferiores para conseguirlo.

Cada tipo de dato multimedia se transmite usando un flujo propio, es decir, si se están transmitiendo a la vez audio y vídeo, el audio se enviará usando una sesión RTP y el vídeo por otra, mediante puertos y direcciones distintas en UDP.

2.3.1. Formato

Los mensajes de RTP se estructuran de forma que se puedan identificar con facilidad en el conjunto de la sesión y entre sesiones. Así, cada uno está marcado con un identificador propio de cada sesión, marcas temporales, número de paquete, fuentes del paquete, etc. Esto es importante ya que se trata de comunicación en tiempo real, y si no se consigue la ordenación correcta en el momento adecuado en el destino esta pierde sentido. La descripción detallada de los mensajes RTP se encuentra en el Anexo C.

2.3.2. RTCP

El protocolo de control RTP [12] se basa en la transmisión de paquetes de control a todos los participantes de la sesión, usando el mismo método de distribución que los paquetes de datos. Tiene tres funciones obligatorias:

- Da información de la calidad del servicio, congestión o fallos en la distribución.
- Contiene un identificador constante de la fuente RTP en caso de que surja algún error.
- Requiere que todos los participantes envíen paquetes RTCP, por lo que todos ellos pueden comprobar cuántos son. Además, sirve para determinar la tasa de envío.

Y una cuarta función opcional:

- Ofrece un mínimo de información de control en sesiones que no tengan un registro de los miembros.

2.3.2.1. Formato

El formato del paquete depende del tipo del paquete:

- *SR* (*Sender Report*, Informe de Emisor): información estadística de la información enviada y recibida por parte de los emisores activos.
- *RR* (*Receiver Report*, Informe de Receptor): información estadística de participantes que no son emisores activos.
- *SDES* (*Source Description*, Descriptor de Fuente): CNAME, NAME, EMAIL, etc.
- *BYE* (Adios): indica la finalización de la participación.
- *APP*: funciones específicas de aplicación.

Los paquetes se procesan de manera independiente y no hay requisitos en cuanto al orden, pero los SR y RR deben enviarse de manera que maximicen la resolución de las estadísticas, los nuevos receptores deben recibir el CNAME de la fuente lo antes posible, el número de tipos de paquetes que aparezcan en el paquete compuesto debe estar limitado para aumentar la probabilidad de éxito de validación. En todos ellos se indican siempre algunas características como la versión, longitud del paquete, número de fuentes de las que se hace el informe, etc. Más información en el Anexo C.

2.3.3. Sistemas Intermedios

RTP cuenta con unos sistemas intermedios llamados *mixers* (mezcladores) y *translators* (traductores) que procesan paquetes RTP y RTCP. Se ha demostrado mediante experimentación con aplicaciones de audio y vídeo en Internet que su uso es necesario. Se utilizan para conectar dos o más *clouds* a nivel de transporte. Un sistema puede actuar como *mixer* o *translator* para una serie de sesiones RTP, aunque cada una se tratará como una entidad lógica separada. Además, todos los puntos RTP finales que se comuniquen a

través de alguno de estos sistemas comparten un mismo espacio SSRC y, por tanto, su identificador debe ser único entre ellos.

Para evitar bucles:

- Cada *cloud* que participe en una sesión RTP conectada por uno de estos elementos debe diferenciarse por lo menos en el protocolo, la dirección o el puerto, o estar aislada del resto a nivel de red.
- No puede haber varios *mixers* o *translators* conectados en paralelo, a no ser que arreglen el reparto de las fuentes.

2.3.3.1. Traductores

Trata el flujo de cada fuente de manera independiente reenviando los paquetes sin afectar al SSRC, lo que permite al receptor identificar el origen de cada uno. Algunos cambian la codificación o añaden información SDES y, por tanto, el *payload* y *timestamp*. En caso de que cambien el *payload* deben modificar los paquetes SR o RR consecuentemente, de forma que muestre las mismas características que los datos al ser recibidos.

- *SR* (*Sender Information*, Emisor de Información):
  El SSRC se mantiene igual, pero la información del emisor debe cambiarse si lo requiere. Si cambia la codificación, debe cambiar el "sender's byte count". Si combina varios paquetes de datos en uno, debe cambiar el "sender's packet count". Si cambia la frecuencia del *timestamp*, debe cambiar el "RTP timestamp".
- *SR/RR reception report blocks*:
  El traductor reenvía los informes de recepción de una *cloud* a otra en sentido opuesto a los datos dejando el SSRC intacto. Si combina varios paquetes de datos, deberá ajustar los números de secuencia y realizar las operaciones opuestas para los campos de pérdida y "extended last sequence number".
- *SDES*:
  No suelen aplicarse cambios, a menos que haya alguna necesidad por ancho de banda. Es necesario que se envíen los CNAMEs.
- Los paquetes BYE y APP se envían sin alterar.

2.3.3.2. Mezcladores

Combina los flujos de las distintas fuentes en uno solo que luego envía sincronizando los diferentes tiempos de cada uno, que por lo general son distintos. Todos los paquetes reenviados por el *mixer* se identificarán con su SSRC y llevarán una lista de los SSRC de las fuentes originales. La mezcla de todos los flujos en uno solo, incluso si hay varias fuentes, permite reducir las necesidades de ancho de banda. Sin embargo, el receptor pierde todo el control sobre cada una de las fuentes, lo que se puede solucionar con mezcladores multimedia.

Como produce un nuevo flujo, tiene que generar sus propios paquetes SR y RR:

- *SR sender information*:

  No reenvía la información de las fuentes porque se pierden al combinarse. Genera un paquete particular con la misma dirección que los que ha mezclado.
- *SR/RR reception report blocks*:

  El mezclador crea su propio informe de recepción para las fuentes de cada cloud y las envía a la propia cloud.
- *SDES*:

  Normalmente no cambia la información SDES recibida, aunque puede hacerlo por necesidad de ancho de banda, manteniendo siempre los CNAMEs y enviando el suyo propio.
- Los paquetes BYE y APP se envían sin alterar.

Varios mezcladores pueden estar conectados en cascada, por lo que los paquetes que reciba cada uno, pueden haber sido ya mezclados e incluir una lista CSRC. En este caso, el segundo mezclador deberá crear su propia lista CSRC para los paquetes de salida, usando los identificadores CSRC que le llegan ya mezclados.

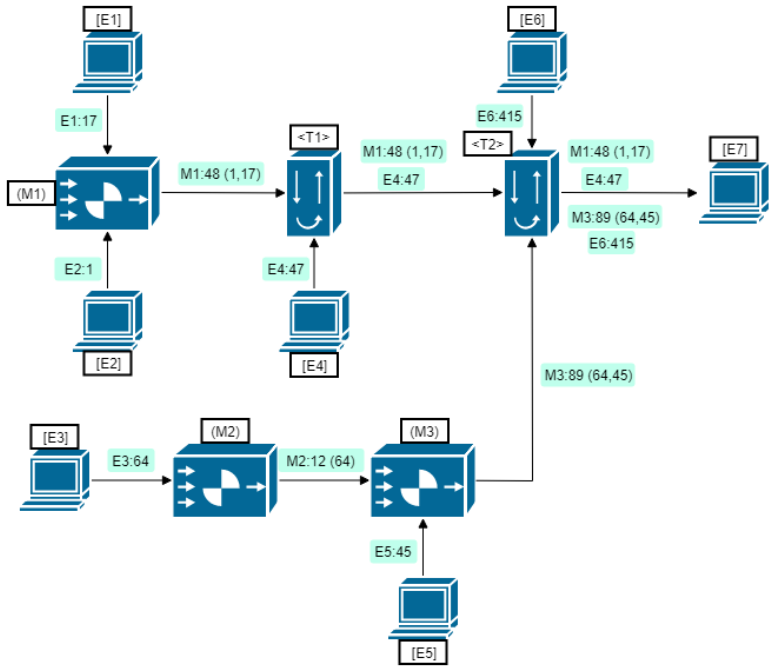

Fig. 6 Esquema de sistemas intermedios en RTP [12]

3. IMPLEMENTACIÓN DEL SISTEMA

Este capítulo trata el desarrollo del sistema que se elabora. Se especifican el diseño y las herramientas a utilizar que implementen los protocolos descritos en el capítulo anterior.

3.1. Descripción del sistema

En las prácticas existentes actualmente tan solo se contempla la existencia de los terminales y un servidor *proxy*/registrador entre ellos. Esto hace que algunos de los elementos principales de Internet, como DNS, no sean tratados. En el contexto del laboratorio que se describe en este trabajo existen, sin embargo, tres elementos fundamentales: el servidor DNS, el servidor *proxy*/registrador y la central telefónica. En el diseño todos ellos se han dispuesto en un mismo ordenador por comodidad, aunque cada uno puede operar por separado. La estructura del sistema es la siguiente:

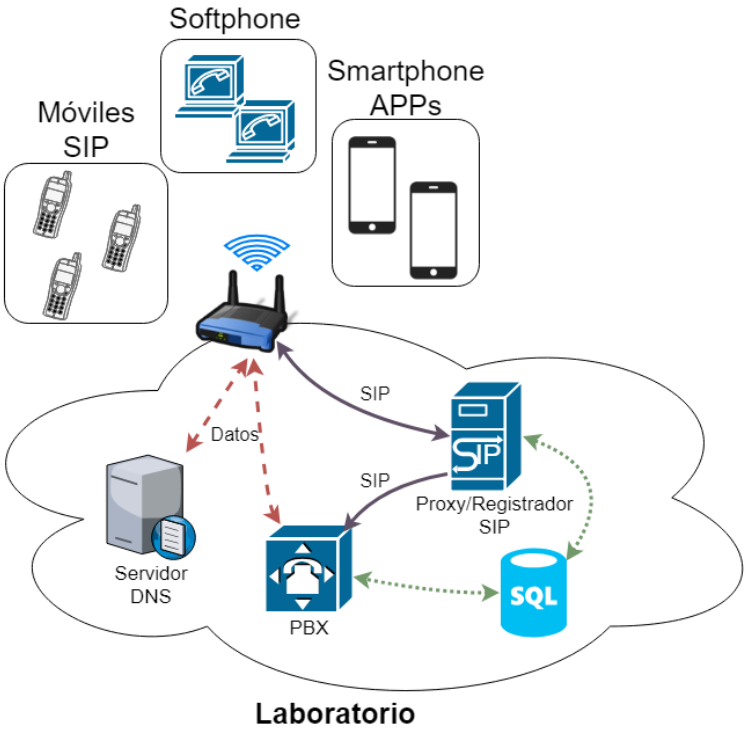

Fig. 7 Arquitectura del sistema

Los terminales móviles se pueden conectar mediante *Wi-Fi* a la red del laboratorio, en la que ya se encuentran los ordenadores usados en las prácticas.

El servidor de DNS, establecido usando el programa BIND, asociará las direcciones IP de cada uno de los dispositivos a un nombre en el dominio en el que se encuentran, en este caso llamado “sip.uc3m.es”. Todo el tráfico SIP pasa por el servidor *proxy* que cumple también la función de registrar a los usuarios. La central telefónica es la encargada de la trasferencia de datos, mantenimiento de llamadas, conferencias, etc. Para realizar

la tarea de *proxy*/registrador se utiliza Kamailio, un *software* que, entre otras cosas, permite gestionar varias instancias de Asterisk, el cual hará la función de central telefónica. Tanto Asterisk como Kamailio acceden a una misma base de datos SQL en la que se encuentran todos los usuarios incluyendo ciertas características, como direcciones IP, puertos, contraseñas o permisos. Por tanto, el sistema quedará de la siguiente manera:

Fig. 8 Arquitectura del sistema con *software* empleado

La instalación de cada uno de los programas se puede encontrar en el Anexo D.

3.2. BIND

BIND es un software creado a principios de los 80 en la universidad de California, que permite publicar en Internet la información DNS propia y resolver peticiones DNS. Es el más usado en Internet y el estándar de facto. Se trata de una herramienta *open source* que puede ser modificada libremente para añadir alguna funcionalidad que no esté presente de base.

3.2.1. Configuración

Empezamos añadiendo la lista de *hosts* y usuarios cuya actividad vamos a permitir.

```
$ vi /etc/bind/named.conf.options
```

Y creamos una lista *Access Control List* (acl) con las direcciones que queramos añadir como de confianza. Las direcciones estarán basadas en "172.1.2.0/24" y el nombre de dominio que asociaremos será "sip.uc3m.es":

```
acl "trusted" {
        172.1.2.99;     # servidor 1 - primario
        172.1.2.100;    # servidor 2 – secundario opcional

        172.1.2.1;      # host1
        …
        172.1.2.20;     # host20

};
```

En el bloque "options" establecemos los siguientes valores para que únicamente nuestro propio servidor resuelva la consulta:

```
        recursion yes;                  # permite recursión en las peticiones
        allow-recursion { trusted; };   # permite recursión de los clientes "trusted"
        listen-on { 172.1.2.99; };      # IP del servidor 1 - escuchar en red privada
        allow-transfer { none; };       # no se copia la información del servidor

        forwarders {
                8.8.8.8;
                8.8.4.4;
        };
```

A continuación editamos "named.conf.local", donde especificamos las zonas *forward* y *reverse*:

```
$ vi /etc/bind/named.conf.local
```

Añadimos:

```
zone "sip.uc3m.es" {
        type master;
        file "/etc/bind/zones/db.sip.uc3m.es";          # ruta a los archivos
        allow-transfer { 172.1.2.100; };                # IP del servidor 2
};

zone "2.1.172.in-addr.arpa" {
        type master;
        file "/etc/bind/zones/db.172.1.2";      # 172.1.2.0/24 subnet
        allow-transfer { 172.1.2.100; };        # IP del servidor 2
};
```

Lo siguiente será crear las zonas donde se establecerá la asociación DNS, a las que recurrirá el servidor. Según hemos indicado en “named.conf.local”, estas estarán en “/etc/bind/zones”, por lo que creamos el directorio:

```
$ mkdir /etc/bind/zones
```

Tomando como referencia el archivo “db.local”, lo copiamos en el escritorio que hemos creado, y lo modificamos para adaptarlo a nuestras necesidades:

```
$ cd /etc/bind/zones
$ cp ../db.local ./db.sip.uc3m.es
$ vi db.sip.uc3m.es
```

Deberá quedar de forma parecida a esta:

```
;
; BIND data file for local loopback interface
;
$TTL 604800
@       IN      SOA     ns1.sip.uc3m.es. admin.sip.uc3m.es. (
                        20180309        ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
;                       name    servers - NS records
                        IN      NS              ns1.sip.uc3m.es.
                        IN      NS              ns2.sip.uc3m.es.

; name servers - A records
sip.uc3m.es.                    IN      A       172.1.2.99
ns1.sip.uc3m.es.                IN      A       172.1.2.99
ns2.sip.uc3m.es.                IN      A       172.1.2.100

; 172.1.2.0/24 - A records
host1.sip.uc3m.es.              IN      A       172.1.2.1
…
host20.sip.uc3m.es.             IN      A       172.1.2.20

; SRV record
;srv.prot.name          ttl     class   rr      pri     weight port    target
_sip.sip.uc3m.es.               IN      SRV     10      20      5060    ns1.sip.uc3m.es.
```

De esta forma en SOA:

- *Serial*: indica la "versión". Tiene el formato AAAAMMDDXX, donde XX indica el número de cambios hechos ese día. Debe cambiarse cada vez que se altera el archivo.
- *Refresh*: tiempo en segundos que tardará el servidor secundario en recargar la zona del principal.
- *Retry*: tiempo en segundos que esperará el secundario para recargar si el *Refresh* falla.
- *Expire*: tiempo en segundos durante el cual el servidor secundario podrá dar una respuesta autorizada.
- *Negative Cache TTL*: valor por defecto del TTL para los RRs en los que no haya sido definido.

A los servidores ns1 y ns2 les asignamos respectivamente las direcciones 172.1.2.99 y 172.1.2.100; y a los *hosts* las direcciones entre 172.1.2.1 y 172.1.2.20.

El registro SRV:

Tiene una estructura:

```
;srv.prot.name          ttl       class   rr      pri     weight port     target
```

- *srv.prot.name*:
  - "srv" es el nombre simbólico del servicio precedido de un guion bajo "_" para evitar colisiones con otras etiquetas de DNS.
  - "prot" es el nombre simbólico del protocolo igualmente precedido de un guion bajo para evitar colisiones. Normalmente utiliza "_tcp" o "_udp".
  - "name" es el nombre del dominio al que hace referencia el RR.
- *ttl*: tiempo de vida.
- *class*: la clase del recurso, que será siempre "IN".
- *pri*: prioridad que tiene el host objetivo. Cuanto menor sea el valor, mayor será la preferencia.
- *weight*: el peso. Es un mecanismo de selección del servidor cuando existe la misma prioridad. Cuanto mayor sea, mayor será la probabilidad de ser seleccionado.
- *port*: puerto que usará el *host* del servicio.
- *target*: el nombre de dominio del *host* objetivo.

Posteriormente, crearemos la zona inversa, que se utilizará para buscar el nombre correspondiente a la dirección IP solicitada.

De forma similar a la anterior, tomaremos como base el archivo db.127, que copiaremos en el directorio "/zones":

```
$ cp /etc/db.127 ./db.172.1.2
```

Y lo editamos de la misma manera:

```
$ vi db.172.1.2
```

Para que quede de acuerdo con nuestras necesidades:

```
;
; BIND reverse data file for local loopback interface
;
$TTL 604800
@     IN      SOA   ns1.sip.uc3m.es. admin.sip.uc3m.es. (
                        2018030902          ; Serial
                        604800              ; Refresh
                        86400               ; Retry
                        2419200             ; Expire
                        604800 )            ; Negative Cache TTL
;
; name servers - NS records
      IN      NS    ns1.sip.uc3m.es.
      IN      NS    ns2.sip.uc3m.es.

; PTR Records
99    IN      PTR   ns1.sip.uc3m.es.      ; 172.1.2.99
100   IN      PTR   ns2.sip.uc3m.es.      ; 172.1.2.100
1     IN      PTR   host1.sip.uc3m.es.    ; 172.1.2.1
…
20    IN      PTR   host20.sip.uc3m.es.   ; 172.1.2.20
```

Es imprescindible que ambos valores del "Serial*"* sean iguales.

Los registros PTR que añadimos corresponden a todos los servidores cuyas IPs se encuentre en la zona. Se incluyen también los *hosts*. La primera columna corresponde al último octeto de cada IP, que identifica a cada elemento de manera particular.

Una vez hayamos terminado de modificar esto archivos tendremos que comprobar que no tengan ningún error, utilizando el comando "named-checkzone", y deberemos reiniciar BIND para que se apliquen los cambios realizados.

```
$ named-checkzone sip.uc3m.es db.sip.uc3m.es

$ named-checkzone 172.1.2.in-addr.arpa db.172.1.2

$ service bind9 restart
```

Para que pueda ser usado al reiniciar el equipo conviene editar el archivo "tail":

```
$ vi /etc/resolvconf/resolv.conf.d/tail
```

Y añadir lo siguiente:

```
search sip.uc3m.es          # nombre del dominio
nameserver 172.1.2.99       # IP del servidor principal
nameserver 172.1.2.100      # IP del segundo servidor si se utiliza
```

Guardamos y ejecutamos:

```
$ resolvconf -u
```

3.3. Asterisk

Asterisk es un *open source framework* para creación de aplicaciones de comunicación. Proporciona funcionalidad de PBX (*Private Branch Exchange*, Ramal Privado de Conmutación) y, por tanto, permite establecer servidores de VoIP, conferencias y otras estructuras, hasta organizar un sistema telefónico completo.

Asterisk soporta la conexión con los sistemas telefónicos tradicionales, así como con los protocolos usados en VoIP. Presenta una estructura modular para poder activar y desactivar los módulos según las necesidades para permitir funciones como correo de voz, grabación de llamada, llamadas en espera, etc.

- *Channel Drivers* (Controladores de Canal)

  Comunican con dispositivos externos a Asterisk y traducen la señalización o protocolo al núcleo.

- *Dialplan Applications* (Aplicaciones de Plan de Marcación)

  Las aplicaciones ofrecen funcionalidades de llamada al sistema. Puede responder llamadas, colgar o poner en cola llamadas.

- *Dialplan Functions* (Funciones de Plan de Marcación)

  Funciones que manipulan algún aspecto de la configuración de la llamada.

- *Resources* (Recursos)

  Proporciona recursos a los módulos de Asterisk.

- *CODECs*

  Módulo para codificar y decodificar audio o vídeo para que utilicen menos ancho de banda. Traducen audio entre códecs de audio y tipos de *payload* usados en distintos dispositivos.

- *File Format Drivers* (Controladores de Formato de Archivo)

  Guardan información multimedia en el disco con un formato específico y reconvierten esos archivos de nuevo a al flujo de la red.

- *Call Detail Record (CDR) Drivers* (Controladores de Informes de Detalles de Llamada)

  Escriben registros de las llamadas al disco o a una base de datos.

- *Call Event Log (CEL) Drivers* (Controladores de Registros de Eventos de Llamada)

  Similar al CDR pero ofrece más información de lo que ha ocurrido en Asterisk durante una llamada.

- *Bridge Drivers* (Controladores Puente)

  Usado para unir multimedia de una llamada entre los participantes.

3.3.1. Configuración

Existen cuatro archivos que deberemos configurar para conseguir la funcionalidad deseada en cuanto a llamadas y conferencias:

3.3.1.1. /etc/asterisk/sip.conf

Contiene parámetros generales del contexto de SIP, donde se pueden especificar direcciones IP, puertos, protocolos de transporte, planes de llamada, entornos de usuarios con características específicas, etc.

Guardar en *cache* amigos en tiempo real mediante una lista interna:

```
rtcachefriends=yes
```

Podemos indicar las direcciones y puertos de escucha, aunque por defecto debería escuchar todas las direcciones y el puerto 5080:

```
udpbindaddr=172.1.2.99:5080

tcpbindaddr=172.1.2.99:5080
```

El idioma para las configuraciones de los usuarios. Para seleccionar español usamos el código del idioma “es”:

```
language=es
```

3.3.1.2. /etc/asterisk/asterisk.conf

Junto a la opción anterior es conveniente establecer el idioma por defecto:

```
defaultlanguage = es
```

3.3.1.3. /etc/asterisk/extensions.conf

Aquí crearemos los planes de llamada y podremos indicar qué hacer en cada situación, como activar el buzón de voz si no ha habido respuesta al cabo de unos segundos o reproducir avisos con cierta información. Estos se escucharán en el idioma indicado anteriormente.

Creamos la extensión "practicas" que incluye la funcione que permiten crear conferencias o unirse a una, definida más adelante. El funcionamiento de una llamada normal es simple, intenta establecer una llamada a un número de cuatro cifras que empiece por "00", si tiene éxito este recibirá la llamada, si por alguna razón no está disponible se colgará:

```
[practicas]
include => conferencia
exten => _00XX,1,NoOp(llamada interna)
exten => _00XX,n,Dial(SIP/${EXTEN})
exten => _00XX,n,Hangup
```

La extensión "conferencia" es la que permite tanto crear como unirse a una conferencia a partir del número marcado, que será el que identifique a la conferencia y a partir del cual los demás usuarios podrán unirse a ella. Se ha configurado para que todas las conferencias empiecen por el número 8 y tengan cuatro dígitos, por lo que todas estarán comprendidas entre el 8000 y el 8999:

1. El usuario tendrá que marcar ese número como en una llamada normal, comenzando siempre por 8.
2. El sistema comprobará que existe esa conferencia. Si ya existe, pasaremos directamente a la última línea, marcada como 10.
3. Se crea la conferencia con el identificador que el usuario ha marcado.
4. Añadirá al usuario y le asignará las propiedades del usuario y conferencia elegidos.

```
[conferencia]
exten => _8XXX,1,NoOp()
same => n,Set(CONFERENCENUM=${EXTEN})
```

```
same => n,GotoIf($[${GROUP_COUNT(${CONFERENCENUM}@conference)} ==
0] ? 10)  ; si existe vamos a 10
same => n,Set(GROUP(conference)=${CONFERENCENUM})   ; si no existe la
creamos
same =>
10,ConfBridge(${CONFERENCENUM},bridge_practicas,user_practicas,menu_practi
cas)
```

3.3.1.4. /etc/asterisk/confbridge.conf

En este archivo se describen las posibles configuraciones de la conferencia, así como la de los usuarios.

El perfil de usuario se designa mediante el tipo "type=user". Este podrá ser administrador y tener algunos privilegios, pero en principio no lo utilizaremos. A su vez se pueden incluir notificaciones a los miembros que ya participen en la conferencia mediante "announce_user_count_all=yes". Este valor puede ser "yes", "no" o un número a partir del cual se hará el anuncio. Además, se puede establecer una contraseña para acceder a la conversación mediante DTMF (*Dual-Tone Multi-Frequency signaling*, Marcación por Tonos) aunque no lo implementaremos:

```
[user_practicas]
type=user
;admin=yes     ; Sets if the user is an admin or not. Off by default.
announce_user_count_all=yes

;pin=1234     ; Sets if this user must enter a PIN number before entering
               ; the conference.  The PIN will be prompted for.
```

Un *bridge* (puente) es la construcción mediante la que se transmite la información entre los canales, que representan las rutas entre los dispositivos. Se pueden definir los usuarios máximos en la conferencia, sonidos para distintos eventos, frecuencia de muestreo, etc. Por ahora solamente estableceremos el idioma:

```
[bridge_practicas]
type=bridge
language=es
```

Finalmente, el menú da la posibilidad de asignar funciones especiales en el teclado de los usuarios. El botón 2 servirá para silenciar el micrófono propio y que los demás usuarios no lo escuchen. Los botones 1 y 3 se usarán para subir y bajar el volumen al que se escucha a los demás:

```
[menu_practicas]
type=menu
; La siguiente linea inidica qué hace cada numero que se pulsa
```

```
*=playback_and_continue(press&digits/1&confbridge-dec-list-vol-
out&press&digits/3&confbridge-mute-out&confbridge-menu-exit-
in&press&digits/3&confbridge-inc-list-vol-out&press&digits/0)
*1=decrease_listening_volume
1=decrease_listening_volume
*2=toggle_mute
2=toggle_mute
*3=increase_listening_volume
3=increase_listening_volume
*0=no_op
```

3.4. Kamailio

Kamailio es un servidor SIP *open source* que permite gestionar miles de establecimientos de llamada por segundo, por lo que es utilizado para crear plataformas de servicios VoIP y comunicación en tiempo real. Además, puede usarse para escalar otros sistemas como Asterisk o FreeSWITCH.

Se complementa con una serie de módulos que proporciona algunas características como las siguientes:

- Servidor SIP (Registrar, Location, Proxy, Application, Redirect).
- Soporta diversos protocolos (TCP, UDP, SCTP).
- Seguridad basada en TLS.
- Soporte de WebSocket para WebRTC.
- IPv4 e IPv6.
- Mensajería instantánea mediante SIMPLE (Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions).
- Operaciones asíncronas.
- DNS y ENUM (Electronic Number Mapping System, Sistema de Mapeo de Números Electrónicos).
- Balance de carga.
- Soporte de sistemas finales para autenticación y autorización (MySQL, Cassandra, MongoDB, etc.).

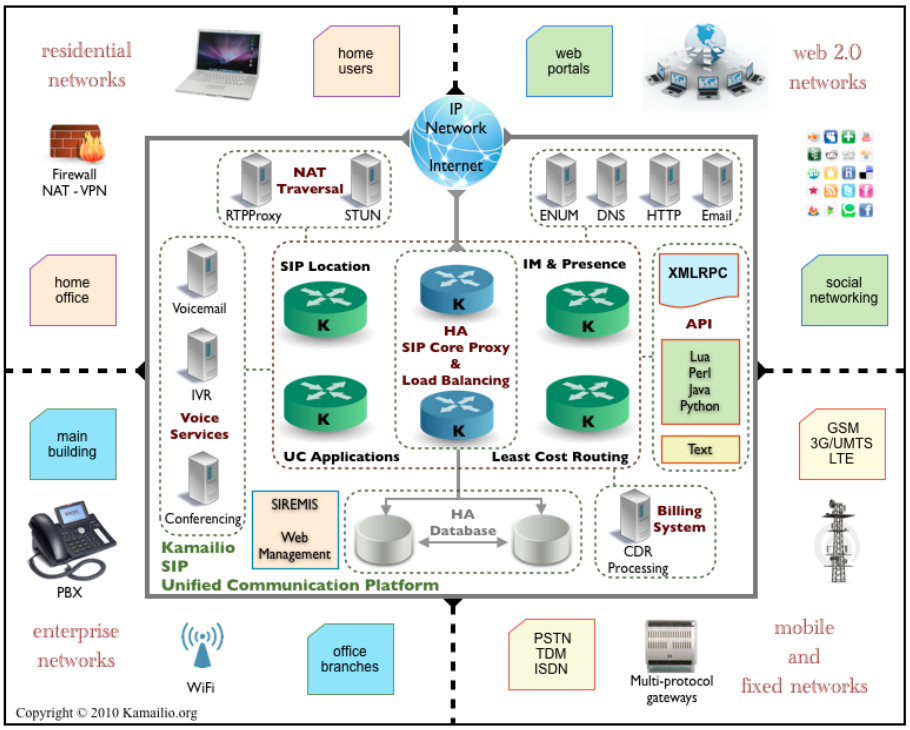

Fig. 9 Estructura de Kamailio [16]

3.4.1. Integración

La integración de Kamailio con Asterisk se realiza para permitir la interacción entre ambos *software* y conseguir de este modo el resultado deseado, en el que Kamailio actúa como *proxy*/*registrar* y Asterisk gestiona el resto de los componentes de las llamadas telefónicas. Para ello tendremos que volver a modificar el archivo "kamailio.cfg". Podemos dividir este proceso en tres partes, tal y como viene estructurado el archivo original:

- Parámetros generales: aquí se definen aspectos como las bases de datos, direcciones IP, nombre de dominio, etc.
- Sección de módulos: donde se pueden añadir diferentes módulos que añadirán las funcionalidades necesarias.
- Lógica de enrutamiento: es la sección en la que se precisa el modo de actuar con las distintas peticiones recibidas.

3.4.1.1. Parámetros generales

Lo primero que habrá que hacer será especificar que se empleará Asterisk de manera similar a cuando lo hicimos con MYSQL, añadiendo en el mismo apartado:

```
#!define WITH_ASTERISK
```

En la parte de “Defined Values” se indican las variables “DBURL” y “DBASTURL” para acceder a las bases de datos de Kamailio y Asterisk:

```
#!ifdef WITH_MYSQL
# - database URL - used to connect to database server by modules such
#       as: auth_db, acc, usrloc, a.s.o.
#!ifndef DBURL
#!define DBURL "mysql://kamailio:kamailiorw@localhost/kamailio"
#!ifdef WITH_ASTERISK
#!define DBASTURL "mysql://asterisk:asterisk_password@localhost/asterisk"
#!endif
#!endif
```

En “Global Parameters” se pueden especificar los alias, direcciónes IP y puertos de escucha:

```
alias="sip.uc3m.es"
listen=172.1.2.99
port=5060
```

Al estar asociado a Asterisk tendremos que especificar en “Custom Parameters” las direcciones y puertos de los servidores donde se alojan:

```
#!ifdef WITH_ASTERISK
asterisk.bindip = "172.1.2.99" desc "Asterisk IP Address"
asterisk.bindport = "5080" desc "Asterisk Port"
kamailio.bindip = "172.1.2.99" desc "Kamailio IP Address"
kamailio.bindport = "5060" desc "Kamailio Port"
#!endif
```

3.4.1.2. Sección de módulos

Cargamos el módulo “uac.so” que añade funcionalidades de *User Agent Client* para transmitir mensajes SIP:

```
#!ifdef WITH_ASTERISK
loadmodule "uac.so"
#!endif
```

El parámetro “rr”, un parte del *header* SIP que guarda un registro de los *proxys* por los que pasa:

```
# ----- rr params -----
# add value to ;lr param to cope with most of the UAs
modparam("rr", "enable_full_lr", 1)
# do not append from tag to the RR (no need for this script)
#!ifdef WITH_ASTERISK
```

```
modparam("rr", "append_fromtag", 1)
#!else
modparam("rr", "append_fromtag", 0)
#!endif
```

Parámetro “auth_db” para autenticación en la base de datos:

```
# ----- auth_db params -----
#!ifdef WITH_AUTH
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "load_credentials", "")

#!ifdef WITH_ASTERISK
modparam("auth_db", "user_column", "name")
modparam("auth_db", "password_column", "sippasswd")
modparam("auth_db", "db_url", DBASTURL)
modparam("auth_db", "version_table", 0)
#!else
modparam("auth_db", "db_url", DBURL)
modparam("auth_db", "password_column", "password")
modparam("auth_db", "use_domain", MULTIDOMAIN)
#!endif
```

3.4.1.3. Lógica de enrutamiento

En la función “route[REGISTRAR]” añadimos para gestionar los registros SIP:

```
#!ifdef WITH_ASTERISK
	route(REGFWD);
#!endif
```

En la función “route[LOCATION]” usada para los servicios de localización:

```
#!ifdef WITH_ASTERISK
	if(is_method("INVITE") && (!route(FROMASTERISK))) {
		# if new call from out there - send to Asterisk
		# - non-INVITE request are routed directly by Kamailio
		# - traffic from Asterisk is routed also directy by Kamailio
		route(TOASTERISK);
		exit;
	}
#!endif
```

La función “route[AUTH]” la modificamos para no filtrar el tráfico de Asterisk, que consideramos de confianza:

```
#!ifdef WITH_ASTERISK
```

```
        # do not auth traffic from Asterisk - trusted!
        if(route(FROMASTERISK))
                return;
#!endif

[. . .]

      if (is_method("REGISTER") || from_uri==myself) {
                        # authenticate requests

     #!ifdef WITH_ASTERISK
                        if (!auth_check("$fd", "sipusers", "1")) {
     #!else
                        if (!auth_check("$fd", "subscriber", "1")) {
     #!endif

[. . .]
```

Por último, crearemos las funciones para enviar el tráfico hacia Asterisk:

```
#!ifdef WITH_ASTERISK
# Test if coming from Asterisk
route[FROMASTERISK] {
        if($si==$sel(cfg_get.asterisk.bindip)
                        && $sp==$sel(cfg_get.asterisk.bindport))
                return 1;
        return -1;
}

# Send to Asterisk
route[TOASTERISK] {
        $du = "sip:" + $sel(cfg_get.asterisk.bindip) + ":"
                        + $sel(cfg_get.asterisk.bindport);
        route(RELAY);
        exit;
}

# Forward REGISTER to Asterisk
route[REGFWD] {
        if(!is_method("REGISTER"))
        {
                return;
        }
        $var(rip) = $sel(cfg_get.asterisk.bindip);
        $uac_req(method)="REGISTER";
        $uac_req(ruri)="sip:" + $var(rip) + ":" + $sel(cfg_get.asterisk.bindport);
        $uac_req(furi)="sip:" + $au + "@" + $var(rip);
        $uac_req(turi)="sip:" + $au + "@" + $var(rip);
        $uac_req(hdrs)="Contact: <sip:" + $au + "@"
                                + $sel(cfg_get.kamailio.bindip)
```

```
                                + ":" + $sel(cfg_get.kamailio.bindport) + ">\r\n";
        if($sel(contact.expires) != $null)
                $uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $sel(contact.expires) +
"\r\n";
        else
                $uac_req(hdrs)= $uac_req(hdrs) + "Expires: " + $hdr(Expires) + "\r\n";
        uac_req_send();
}
#!endif
```

3.5. Creación de usuarios

La manera de añadir nuevos usuarios que puedan utilizar los servicios es a través de la base de datos Asterisk creada anteriormente. Para cada uno se deberán introducir los datos en las columnas correspondientes con el mismo formato que en el archivo "etc/asterisk/sip.conf". Es posible, por tanto, añadirlos directamente en ese fichero, aunque no lo haremos así, ya que buscamos que el registro se realice a través de Kamailio accediendo a la base de datos, y la gestión por parte de Asterisk sea dinámica.

Los datos que deberemos rellenar en la tabla "sipusers" son: *name*, *defaultuser*, *host*, *sippaswd*, *context*. El resto de las columnas se completará automáticamente por Asterisk una vez se haya registrado el usuario.

- *Name*: un nombre de usuario, que será el que actúe como número de teléfono.
- *Defaultuser*: nombre para la autenticación.
- *Host*: estableciéndolo como *dynamic* obligamos al usuario a registrarse.
- *Sippasswd*: contraseña que deberá introducir el usuario en el registro.
- *Context*: es el contexto en el que operará el usuario. En este caso el que hemos creado en "etc/asterisk/extensions.conf".

Adicionalmente podemos definir parámetros como

- *Qualify*: para comprobar regularmente si el usuario se encuentra a rango y el valor de este.
- *Mailbox*: si queremos activar el buzón de voz.
- *Fromdomain*: para hacer llamadas hacia el exterior.
- *Fromuser*: una manera para que te reconozca el servidor. Algunos servidores SIP lo requieren.

En la tabla "sipregs" solamente es necesario el nombre (name) de usuario para el registro.

Hay que tener en cuenta que, si se establece un buzón de voz, es necesario también rellenar los datos correspondientes en la tabla "voicemail":

```
INSERT INTO voicemail(context, mailbox, password) VALUES ('practicas', '0017',
'1234');
```

La consulta que realizaremos en MySQL para añadir a un usuario con estas características será:

```
INSERT INTO sipusers (name, defaultuser, host, sippasswd, context, qualify)
VALUES ('0017', '0017', 'dynamic', '1234', 'practicas', ‘yes’);
INSERT INTO sipregs(name) VALUES('0017');
```

3.6. Configuración de los terminales

3.6.1. Móviles *Wi-Fi*

Los dispositivos disponibles en los laboratorios son móviles VoIP Wi-Fi modelo ZyXEL p-2000w_v2. Empezaremos por encender el teléfono manteniendo la tecla roja. Accedemos al menú y deberemos realizar varios cambios en las opciones. Toda la configuración se puede hacer a través de una interfaz en un ordenador, a la que se accede escribiendo en el buscador de Internet la IP del dispositivo o el *host* correspondiente, como indicamos en BIND. Se nos pedirá un nombre de usuario (zyxeladmin) y contraseña (1234):

1. *Net settings*:
    a. *Network mode*: modo de la red para asignar direcciones IP. El *router* disponible está configurado de manera que se le asignen direcciones fijas, por tanto, seleccionaremos “fixed IP”.
    b. *IP address*: aquí elegimos la dirección IP. Las que se han elegido al configurar DNS estaban en el rango 172.1.2.1-172.1.2.20, y el servidor en 172.1.2.99. Por lo que utilizamos una de ellas, por ejemplo 172.1.2.17
    c. *Subnet mask*: la máscara de red es 255.255.255.0

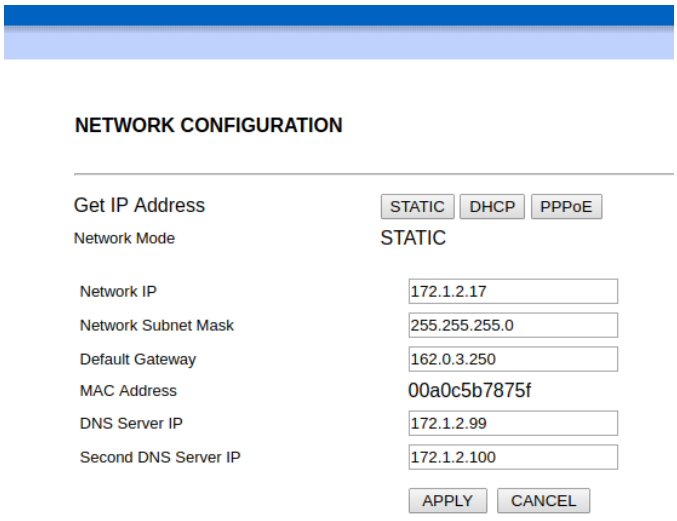

Fig. 10 Configuración de red en móviles ZyXEL

2. *SIP settings*

a. *Registrar*
   i. *Registrar IP*: la dirección IP en la que registraremos al dispositivo. Es posible usar el dominio: sip.uc3m.es
   ii. *Port*: el puerto donde se registrará (5060 por defecto en SIP).
   iii. *Expiry time*: tiempo que durará el registro en segundos. Marcamos 120.

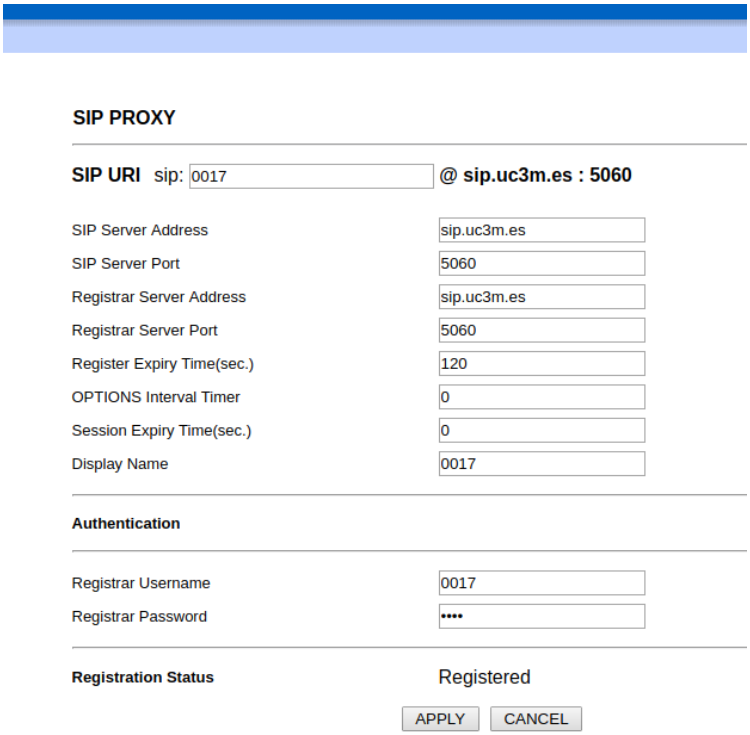

Fig. 11 Configuración *SIP Proxy* en móviles ZyXEL

b. *Outbound proxy*
   i. *Proxy IP*: dirección IP del *proxy* de salida. Marcaremos el mismo.
   ii. *Port*: el puerto también será igual.

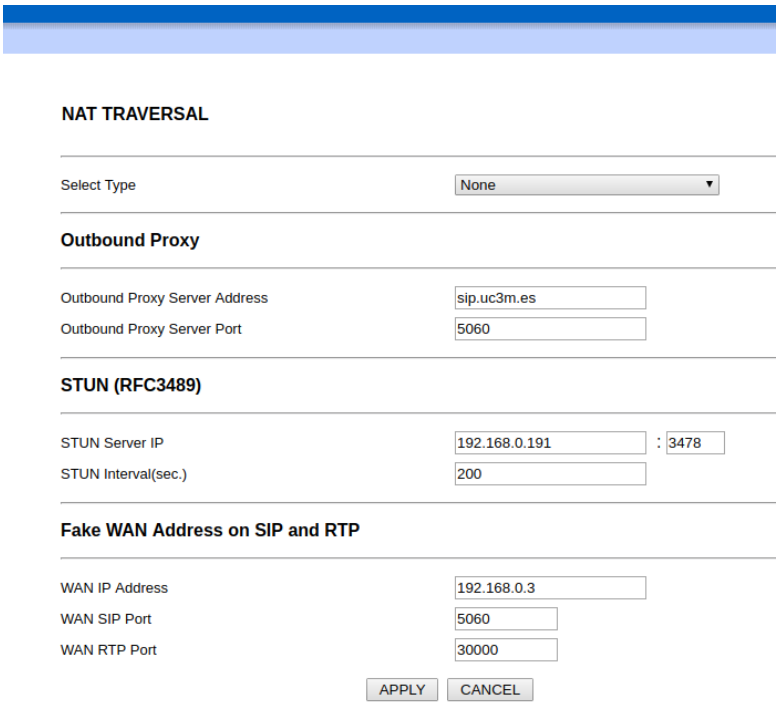

Fig. 12 Configuración *Outbound Proxy* en móviles ZyXEL

c. *Proxy server*
    i. *Proxy IP*: la dirección del servidor *proxy*, igual que el registrar.
    ii. *Port*: el mismo.
d. *User account*
    i. *Phone number*: el número de teléfono con el que se harán las llamadas. Tiene que estar adecuado a la extensión creada en Asterisk, por ejemplo 0017.
    ii. *Username*: nombre de usuario para hacer el registro: 0017.
    iii. *User pwd*: contraseña para registrarse: 1234.

3. *Wireless*
    a. *SSID*: *Service Set Identifier* (Identificador de Conjunto de Servicios) es el nombre con el que se identifica a una red. La que está configurada en el *router* es “practicas_IT.UC3M”.

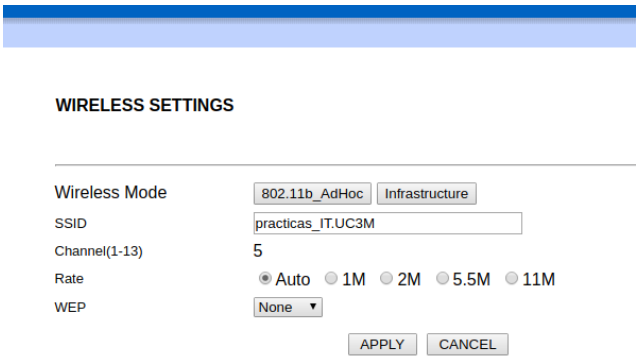

Fig. 13 Configuración *Wireless* en móviles ZyXEL

La opción “phonebook” permite insertar contactos a los que asignaremos un nombre, el número de teléfono, la dirección IP del host al que está asociada, un puerto y el método para contactar con él, que será mediante proxy.

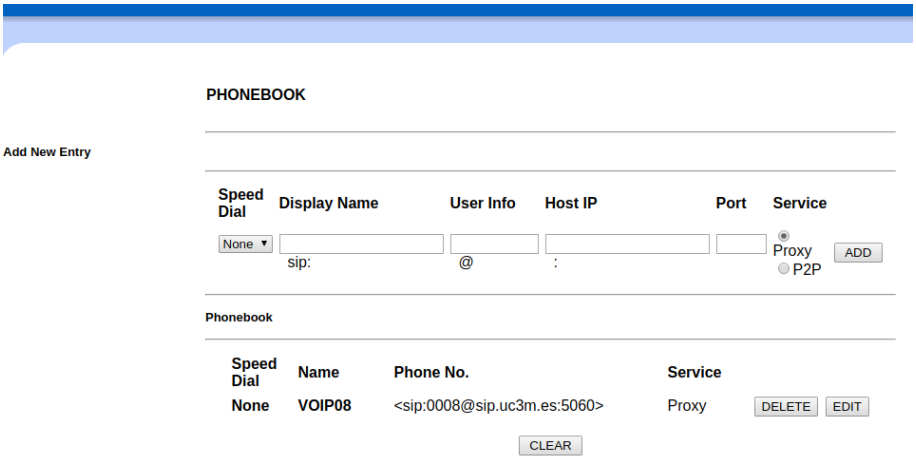

Fig. 14 Agenda de contactos en móviles ZyXEL

3.6.2. Softphone PC

En los ordenadores del departamento de telemática ya se encuentra instalado Ekiga, un *softphone open source* que permite realizer llamadas, conferencias e intercambiar mensajes de texto a través de Internet. Consta de una sencilla interfaz que permite acceder a los contactos, conversaciones o elegir entre distintas cuentas.

Por tanto, el primer paso será crear una de estas cuentas pulsando Editar>Cuentas. Aquí, según las necesidades, se pueden crear distintos tipos, aunque elegiremos una cuenta SIP. Se debe rellenar la misma información: nombre de usuario, servidor de registro y contraseña; y activar la cuenta.

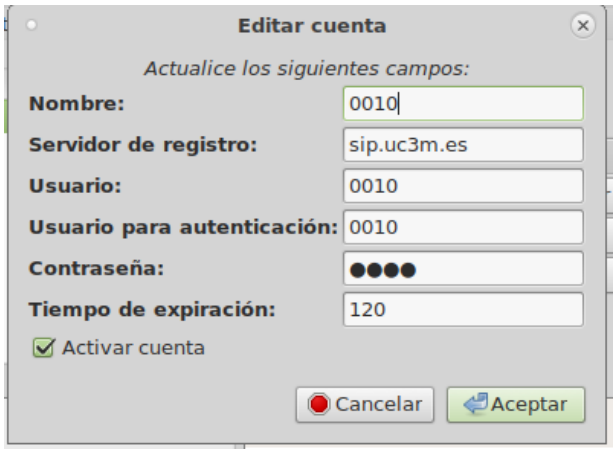

Fig. 15 Edición de una cuenta SIP en Ekiga

Desde este momento se podrán realizar solicitudes INVITE, recibir llamadas, e incluso enviar mensajes de texto. Para ello hay que indicar la URI del destinatario en el campo inferior:

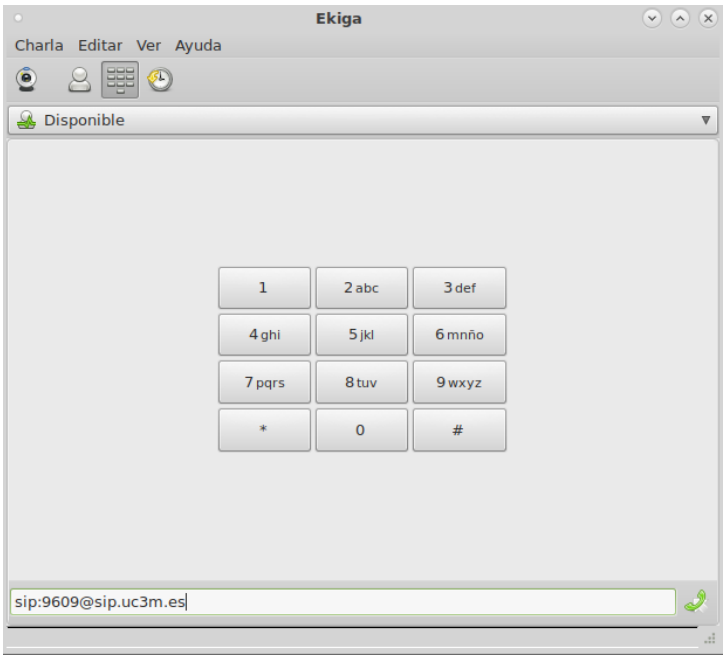

Fig. 16 Vista de la pantalla principal de Ekiga

3.6.3. Smartphone APPs

Para *smartphone* se han probado dos aplicaciones cuyo funcionamiento es similar. En ambas se debe iniciar una cuenta al igual que se ha hecho en Ekiga, introduciendo los datos del usuario y el servidor en el que se vaya a usar. Sin embargo, cada de estas aplicaciones tiene unas características específicas:

- Media5-Fone
  - Es gratis, aunque contiene anuncios
  - Permite observar directamente las trazas SIP transmitidas
  - Buzón de voz integrado
  - Actualizado regularmente
- CSipSimple
  - *Open source*
  - Mensajes de texto
  - Menos actualizado que Media5-Fone

4. PRUEBAS

En este capítulo se realizan una batería de pruebas con las que se podrá determinar si el resultado final del proyecto cumple los objetivos iniciales.

Una vez realizadas todas las configuraciones pertinentes pasaremos a verificar que el funcionamiento es realmente el esperado, y que cumple con los estándares vigentes definidos en los RFCs correspondientes. Para ello se revisarán cada uno de los apartados expuestos anteriormente y se aplicarán comprobaciones específicas para cada uno. Estas comprobaciones consistirán en:

- Identificación de usuarios y servidores VoIP mediante URI SIP.
- Registro de los terminales en el servidor.
- Llamada entre dos terminales y comprobación de las posibles respuestas.
- Conferencia entre dos o más usuarios.

Así pues, habrá que ejecutar los comandos correspondientes para iniciar Kamailio y Asterisk:

```
$ /etc/init.d/kamailio start

$ /etc/init.d/asterisk start
```

Asterisk proporciona una consola desde la que se puede comprobar información relativa a su estado, como número de usuarios conectados, llamadas realizadas, conferencias activas, etc. Se accede a esta vista mediante el comando:

```
$ asterisk -vr
```

Para la captura de paquetes se utilizará Wireshark.

4.1. Resolución DNS

El funcionamiento de las direcciones IP asociadas a nombres de dominio puede comprobarse mediante el encendido del terminal, ya que este está configurado de manera que realicen el registro usando la dirección de dominio “sip.uc3m.es”. Por ello, deberán enviar la solicitud correspondiente al servidor que se encarga de su resolución. La petición se envía desde la dirección “172.1.2.17” a la dirección establecida en la configuración del móvil que es “172.1.2.99”:

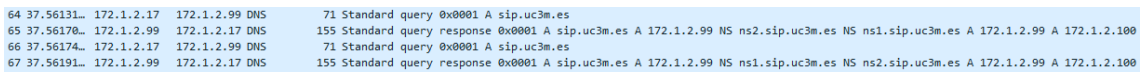

Fig. 17 Mensajes DNS en Wireshark

De manera más detallada:

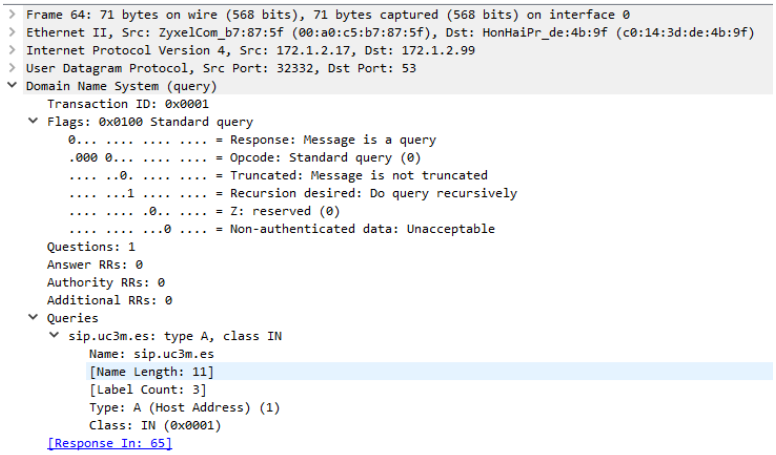

Fig. 18 Solicitud DNS

La petición sigue el formato descrito en el apartado de DNS 2.1.2. En la primera línea (flags) queda indicado que se trata de una solicitud estándar, con recursión y entera. La segunda es la que indica que solamente se trata de una solicitud (Questions: 1). Por ello el resto de los campos del *header* están vacíos. En el apartado de *Queries* se establece que el *QNAME* solicitado es, precisamente, “sip.uc3m.es” y del que se pide su dirección IP (*QTYPE*=A) en el dominio de Internet (*QCLASS*=IN).

La respuesta generada en el servidor sigue el mismo esquema:

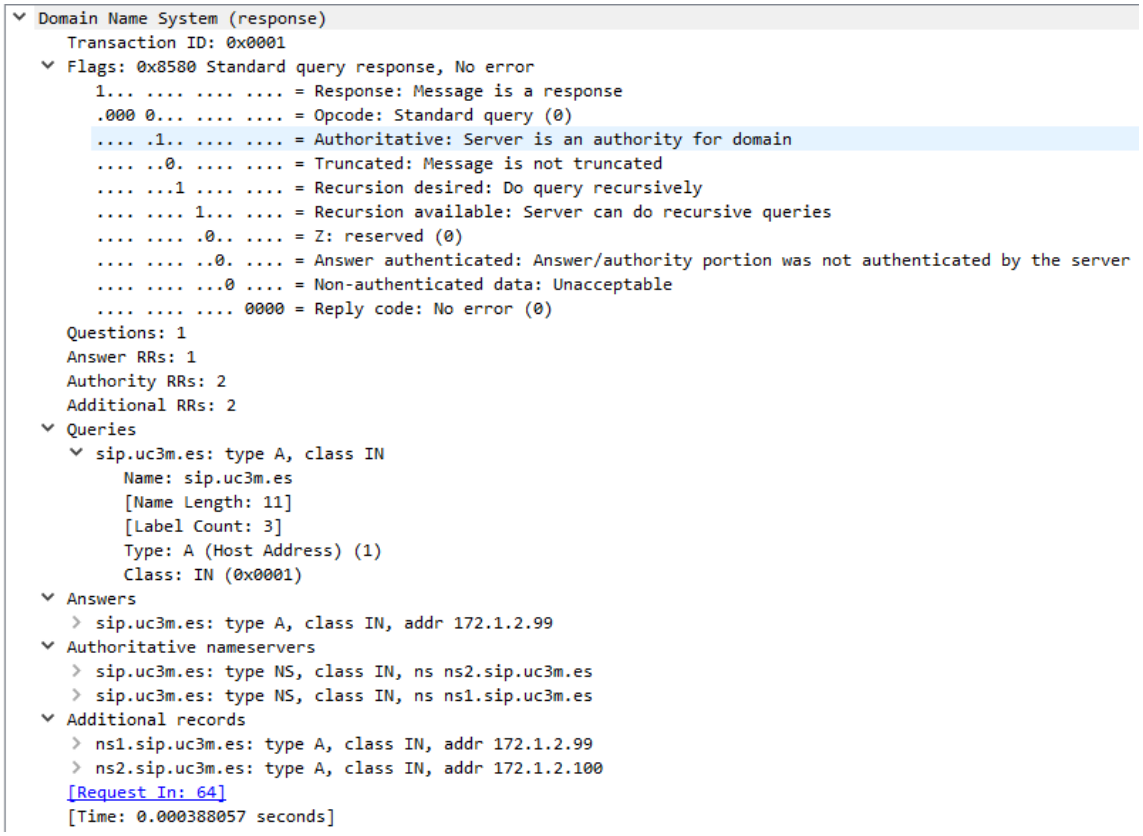

Fig. 19 Respuesta DNS

Se indica que el mensaje es una respuesta estándar, desde un servidor autoritario, que admite recursión y no hay ningún error. Además, en los demás apartados de *header* se indican las respuestas RR, el número de autoridades y el número de registros adicionales:

- *Queries* toma el mismo valor que el de la pregunta original.
- *Answers*: responde con la dirección IP solicitada.
- *Authoritative nameservers*: lista de los servidores autoritarios. Podemos comprobar que coinciden con lo establecido en la configuración de BIND.
- *Additional records*: esto simplemente son las direcciones de los servidores anteriores.

4.2. Registro

Para comprobar que todo funciona correctamente, el teléfono debe pertenecer al contexto de las prácticas y estar registrado. En un principio, el teléfono parte de un estado apagado y es al encenderlo cuando inicia el proceso de registro. Deberá entonces enviar la solicitud de registro al servidor, que comprobará si las credenciales son correctas y actuará en consecuencia, aceptando el registro si los son, o denegándolo en caso de no serlo.

En este caso el teléfono que realiza el registro tiene la dirección 172.1.2.17 y el registrador es Kamailio, con dirección 172.1.2.99 y puerto 5060.

```
52 18.97578… 172.1.2.17  172.1.2.99 SIP      451 Request: REGISTER sip:sip.uc3m.es:5060  (1 binding) |
53 18.97608… 172.1.2.99  172.1.2.17 SIP      501 Status: 401 Unauthorized |
54 18.97816… 172.1.2.99  172.1.2.17 SIP      501 Status: 401 Unauthorized |
55 18.98116… 172.1.2.99  172.1.2.17 SIP      501 Status: 401 Unauthorized |
56 19.05531… 172.1.2.17  172.1.2.99 SIP      630 Request: REGISTER sip:sip.uc3m.es:5060  (1 binding) |
57 19.05585… 172.1.2.17  172.1.2.99 SIP      630 Request: REGISTER sip:sip.uc3m.es:5060  (1 binding) |
58 19.05631… 172.1.2.99  172.1.2.17 SIP      465 Status: 200 OK  (1 binding) |
59 19.05753… 172.1.2.99  172.1.2.17 SIP      465 Status: 200 OK  (1 binding) |
```

Fig. 20 Intercambio de mensajes de registro

De manera más esquemática podemos ver que se ajusta al estándar:

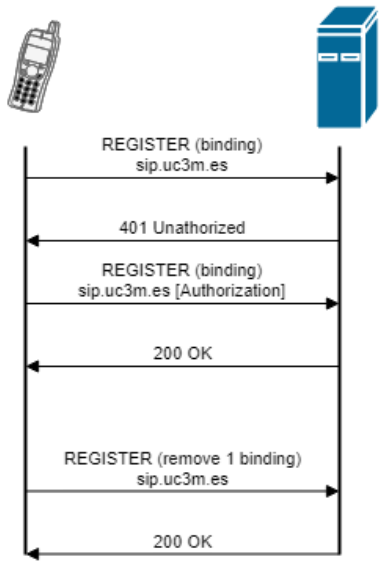

Fig. 21 Flujo de mensajes *REGISTER-UNREGISTER*

Se observa que se envía un primer mensaje de registro al que el servidor responde como "no autorizado". Sin embargo, el mensaje siguiente sí que es aceptado. Esto se debe a que el servidor le ha pedido al UA que envíe la información necesaria para el registro en la base de datos, es decir, el nombre de usuario y contraseña.

De manera más detallada:

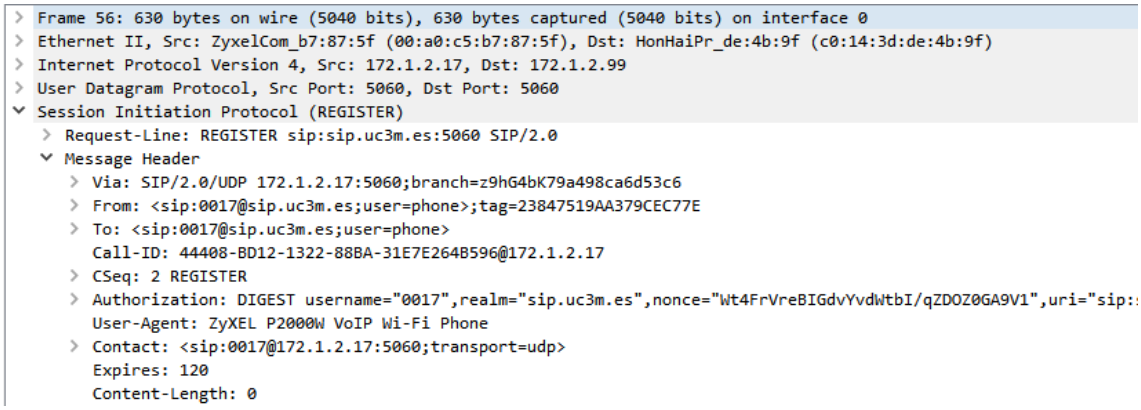

Fig. 22 Mensaje *REGISTER*

En la primera línea determina que la solicitud es de tipo *REGISTER* y se envía a la dirección "sip.uc3m.es" y puerto "5060" con la versión "2.0".

El campo *Via* contiene solamente los datos del emisor ya que no ha habido ningún intermediario. Al ser un registro, *From* y *To* tienen que tener también los datos del que inicia la sesión; además, como es el primer mensaje, en *From* hay que añadirle su correspondiente *tag*. El campo *Contact* establece que las solicitudes se enviarán a la misma dirección que el *From*, y que el tiempo que dura el registro son 120 segundos. Como se ha indicado, es necesario enviar los datos para comprobar que este usuario existe en la base de datos, lo cual se hace en el campo *Authorization*.

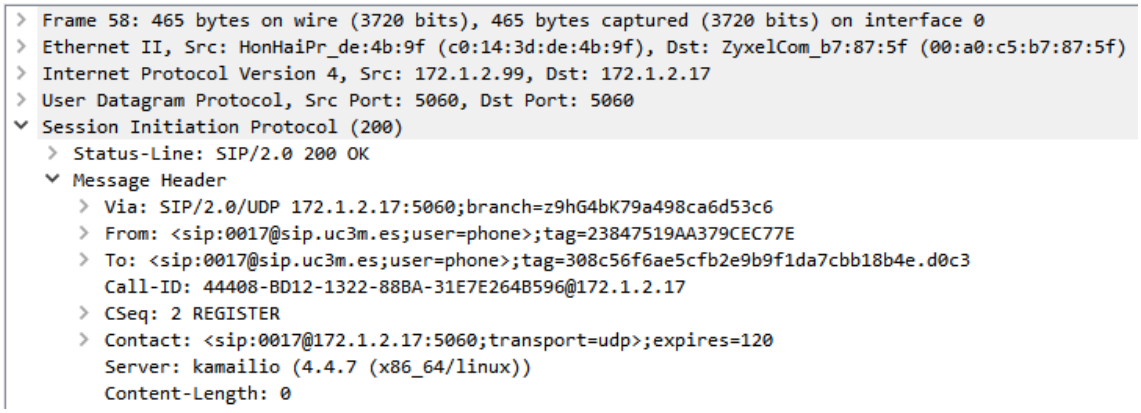

Fig. 23 Mensaje *OK* en registro

Aquí la respuesta dada por el servidor, en la propia respuesta se indica que es Kamailio, es un "200 OK". Los campos del *header* son los mismos que en la solicitud con la excepción del *To*, que añade su propio *tag*.

De manera similar se realiza el *desregistro*:

```
86 31.64224… 172.1.2.17   172.1.2.99 SIP       449 Request: REGISTER sip:sip.uc3m.es:5060  (remove 1 binding) |
87 31.64259… 172.1.2.99   172.1.2.17 SIP       501 Status: 401 Unauthorized |
88 31.74373… 172.1.2.99   172.1.2.17 SIP       501 Status: 401 Unauthorized |
89 31.81421… 172.1.2.17   172.1.2.99 SIP       628 Request: REGISTER sip:sip.uc3m.es:5060  (remove 1 binding) |
90 31.81479… 172.1.2.17   172.1.2.99 SIP       628 Request: REGISTER sip:sip.uc3m.es:5060  (remove 1 binding) |
91 31.81501… 172.1.2.99   172.1.2.17 SIP       402 Status: 200 OK  (0 bindings) |
92 31.81610… 172.1.2.99   172.1.2.17 SIP       402 Status: 200 OK  (0 bindings) |
```

Fig. 24 Intercambio de mensajes de *desregistro*

La manera de finalizar la sesión es con un mensaje con los mismos datos que el de registro, pero cambiando el valor de *Expires* a 0.

```
> Frame 89: 628 bytes on wire (5024 bits), 628 bytes captured (5024 bits) on interface 0
> Ethernet II, Src: ZyxelCom_b7:87:5f (00:a0:c5:b7:87:5f), Dst: HonHaiPr_de:4b:9f (c0:14:3d:de:4b:9f)
> Internet Protocol Version 4, Src: 172.1.2.17, Dst: 172.1.2.99
> User Datagram Protocol, Src Port: 5060, Dst Port: 5060
v Session Initiation Protocol (REGISTER)
  > Request-Line: REGISTER sip:sip.uc3m.es:5060 SIP/2.0
  v Message Header
    > Via: SIP/2.0/UDP 172.1.2.17:5060;branch=z9hG4bKb1e340335470e1
    > From: <sip:0017@sip.uc3m.es;user=phone>;tag=713CCED1782DA6C61D9
    > To: <sip:0017@sip.uc3m.es;user=phone>
      Call-ID: 44408-BD12-1322-88BA-31E7E264B596@172.1.2.17
    > CSeq: 4 REGISTER
    > Authorization: DIGEST username="0017",realm="sip.uc3m.es",nonce="Wt4FulreBI4E5QhgShp90IveXWfutpj/",uri="sip:s
      User-Agent: ZyXEL P2000W VoIP Wi-Fi Phone
    > Contact: <sip:0017@172.1.2.17:5060;transport=udp>
      Expires: 0
      Content-Length: 0
```

Fig. 25 Mensaje *UNREGISTER*

4.3. Llamadas entre dos usuarios

Para demostrar que dos terminales pueden realizar una comunicación VoIP es necesario que ambos hayan completado el paso anterior, el registro. La llamada se inicia mediante una solicitud SIP INVITE en la que el usuario debe especificar el destino, y las condiciones en las que se hace mediante SDP. Tanto esta solicitud como el resto de los mensajes SIP pasan por el *proxy* que las reenvía a sus respectivos destinatarios.

A esta petición de llamada se puede responder de tres maneras distintas que se analizan más adelante:

- OK: el receptor acepta la llamada
- BUSY: el receptor rechaza la llamada
- CANCEL: el emisor termina la llamada antes de que el receptor pueda dar una de las otras respuestas.

En el ejemplo expuesto, el terminal que inicia la llamada es el que tiene la dirección 172.1.2.17 y el llamado es 172.1.2.8, cuyas identidades son 0017 y 0008 respectivamente y se encuentran en el dominio “sip.uc3m.es”. Se observa que ambos extremos pueden alcanzarse y que, por tanto, el sistema funciona.

```
339 176.6281… 172.1.2.17   172.1.2.99 SIP/SDP    761 Request: INVITE sip:0008@sip.uc3m.es:5060 |
341 176.6287… 172.1.2.17   172.1.2.99 SIP/SDP    761 Request: INVITE sip:0008@sip.uc3m.es:5060 |
343 176.8430… 172.1.2.99   172.1.2.17 SIP        510 Status: 407 Proxy Authentication Required |
344 176.8432… 172.1.2.99   172.1.2.17 SIP        510 Status: 407 Proxy Authentication Required |
346 176.8569… 172.1.2.99   172.1.2.17 SIP        510 Status: 407 Proxy Authentication Required |
347 176.8599… 172.1.2.99   172.1.2.17 SIP        510 Status: 407 Proxy Authentication Required |
348 176.9058… 172.1.2.17   172.1.2.99 SIP        413 Request: ACK sip:0008@172.1.2.8:5060 |
349 176.9064… 172.1.2.17   172.1.2.99 SIP        413 Request: ACK sip:0008@172.1.2.8:5060 |
350 176.9589… 172.1.2.17   172.1.2.99 SIP/SDP    948 Request: INVITE sip:0008@172.1.2.8:5060 |
351 176.9595… 172.1.2.17   172.1.2.99 SIP/SDP    948 Request: INVITE sip:0008@172.1.2.8:5060 |
352 176.9601… 172.1.2.99   172.1.2.17 SIP        387 Status: 100 trying -- your call is important to us |
353 176.9603… 172.1.2.99   172.1.2.8  SIP/SDP    924 Request: INVITE sip:0008@172.1.2.8:5060 |
354 176.9623… 172.1.2.99   172.1.2.17 SIP        387 Status: 100 trying -- your call is important to us |
355 177.1467… 172.1.2.99   172.1.2.8  SIP/SDP    924 Request: INVITE sip:0008@172.1.2.8:5060 |
356 177.4355… 172.1.2.99   172.1.2.8  SIP/SDP    924 Request: INVITE sip:0008@172.1.2.8:5060 |
357 177.4544… 172.1.2.99   172.1.2.8  SIP/SDP    924 Request: INVITE sip:0008@172.1.2.8:5060 |
358 178.4355… 172.1.2.99   172.1.2.8  SIP/SDP    924 Request: INVITE sip:0008@172.1.2.8:5060 |
359 178.9899… 172.1.2.8    172.1.2.99 SIP        582 Status: 180 Ringing |
360 178.9902… 172.1.2.99   172.1.2.17 SIP        499 Status: 180 Ringing |
361 178.9904… 172.1.2.8    172.1.2.99 SIP        582 Status: 180 Ringing |
362 178.9906… 172.1.2.99   172.1.2.17 SIP        499 Status: 180 Ringing |
363 178.9927… 172.1.2.99   172.1.2.17 SIP        499 Status: 180 Ringing |
364 178.9941… 172.1.2.99   172.1.2.17 SIP        499 Status: 180 Ringing |
```

Fig. 26 Intercambio de mensajes para establecer una llamada

De forma esquemática y de acuerdo con el estándar:

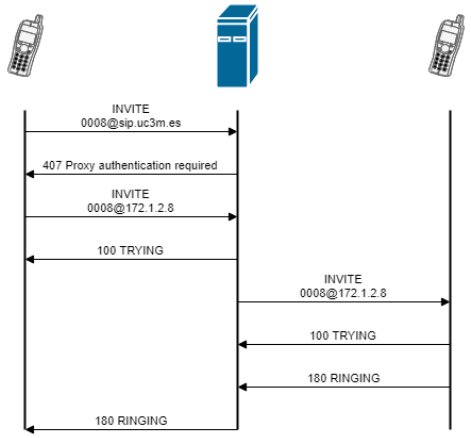

Fig. 27 Flujo de establecimiento de llamada

De manera similar a la anterior, al enviar la solicitud, el *proxy* responde con un mensaje indicando que se requieren los datos del emisor. El mensaje enviado desde el llamante contiene ahora en el *To* la URI del destinatario, que en este caso es “sip:0008@172.1.2.8”. El resto de las componentes del *header* contiene la misma información que en el registro y todos los mensajes reenviados desde el servidor al destinatario son exactamente los mismos, salvo que se le añaden una nueva línea al *Via* con sus datos. A su vez, consta de un cuerpo de mensaje (Message Body) formado por el protocolo SDP. Este identifica al creador de la sesión y ofrece las opciones de transmitir audio usando los códecs G729, PCMU o PCMA:

```
> Frame 350: 948 bytes on wire (7584 bits), 948 bytes captured (7584 bits) on interface 0
> Ethernet II, Src: ZyxelCom_b7:87:5f (00:a0:c5:b7:87:5f), Dst: HonHaiPr_de:4b:9f (c0:14:3d:de:4b:9f)
> Internet Protocol Version 4, Src: 172.1.2.17, Dst: 172.1.2.99
> User Datagram Protocol, Src Port: 5060, Dst Port: 5060
v Session Initiation Protocol (INVITE)
  > Request-Line: INVITE sip:0008@172.1.2.8:5060 SIP/2.0
  v Message Header
    > Via: SIP/2.0/UDP 172.1.2.17:5060;branch=z9hG4bKcbe655aea176cd
    > From: 0017 <sip:0017@sip.uc3m.es;user=phone>;tag=E9BF2338ACB0A7209D59
    > To: <sip:0008@172.1.2.8>
      Call-ID: 21599-BD12-1322-C23A-A4598A19472D@172.1.2.17
    > CSeq: 2 INVITE
    > Proxy-Authorization: DIGEST username="0017",realm="sip.uc3m.es",nonce="WvQIIVr0BvUXa0IY0vKod74MGh4yEHep",uri="sip:0
      Supported: replaces
      Allow: INVITE,OPTIONS,BYE,CANCEL,ACK,SUBSCRIBE,NOTIFY,INFO,REFER
    > Contact: <sip:0017@172.1.2.17:5060;transport=udp>
      Max-Forwards: 70
      User-Agent: ZyXEL P2000W VoIP Wi-Fi Phone
      Content-Type: application/sdp
      Content-Length: 193
  v Message Body
    v Session Description Protocol
        Session Description Protocol Version (v): 0
      > Owner/Creator, Session Id (o): TelogyUnknown0000 38684 38684 IN IP4 172.1.2.17
        Session Name (s): RTP Audio
      > Connection Information (c): IN IP4 172.1.2.17
      > Time Description, active time (t): 0 0
      > Media Description, name and address (m): audio 2070 RTP/AVP 18 0 8
      > Media Attribute (a): rtpmap:18 G729/8000
      > Media Attribute (a): rtpmap:0 PCMU/8000
      > Media Attribute (a): rtpmap:8 PCMA/8000
```

Fig. 28 Mensaje *INVITE*

En este punto pueden ocurrir tres cosas:

4.3.1. OK

En este caso el receptor ha aceptado la llamada con alguna de las condiciones especificadas por el emisor. Requiere, por tanto, de su finalización mediante un intercambio de mensajes SIP *BYE*.

Continuando el esquema anterior de llamada (Fig. 26):

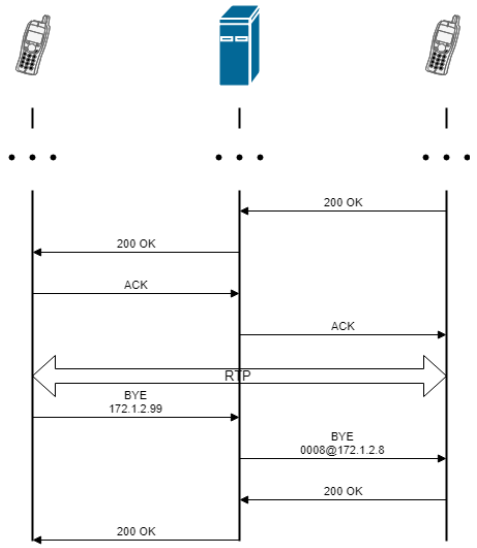

Fig. 29 Flujo de mensajes en una llamada aceptada

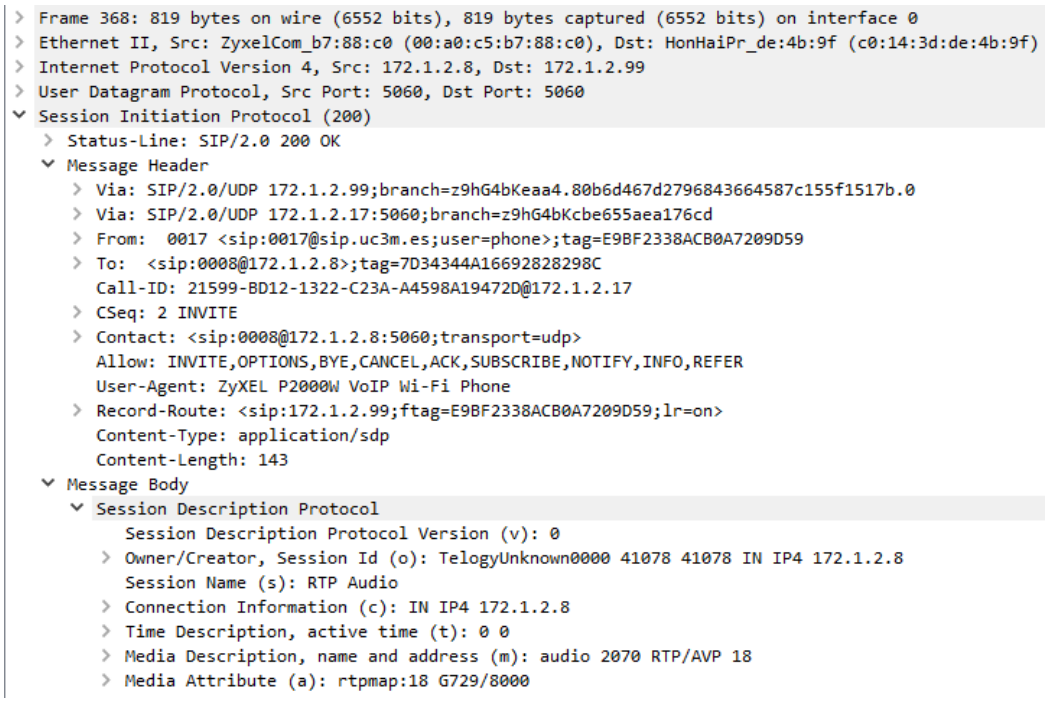

Fig. 30 Mensaje *OK* en *INVITE*

Comprobamos que, al aceptar la solicitud, el cuerpo del mensaje contiene solamente un campo “a”, por tanto, la comunicación RTP se hará usando el códec indicado. La respuesta *OK* se propaga desde el receptor y, una vez le ha llegado al emisor, se manda el *ACK*.

La comunicación termina cuando uno de los dos componentes cuelga mandando un mensaje *BYE*. La confirmación de haberla finalizado se hace una vez que el segundo componente ha recibido la solicitud.

8109	301.1441…	172.1.2.17	172.1.2.99	SIP	421	Request: BYE sip:172.1.2.99 |
8110	301.1444…	172.1.2.99	172.1.2.8	SIP	497	Request: BYE sip:0008@172.1.2.8:5060 |
8111	301.1445…	172.1.2.17	172.1.2.99	SIP	421	Request: BYE sip:172.1.2.99 |
8112	301.1457…	172.1.2.99	172.1.2.8	SIP	497	Request: BYE sip:0008@172.1.2.8:5060 |
8113	301.1917…	172.1.2.8	172.1.2.99	SIP	640	Status: 200 OK |
8114	301.1918…	172.1.2.99	172.1.2.17	SIP	557	Status: 200 OK |
8115	301.1922…	172.1.2.8	172.1.2.99	SIP	640	Status: 200 OK |
8116	301.1934…	172.1.2.99	172.1.2.17	SIP	557	Status: 200 OK |

Fig. 31 Intercambio de mensajes *BYE*

En el siguiente gráfico se muestran la tasa de transmisión y recepción de RTP en bits/s y se señala en verde cuando transmite SIP. Se observa que justo al inicio y al final de la llamada existen estos mensajes y, durante esta, hay alguno que puede estar relacionado con el re-registro o la negociación de alguna opción.

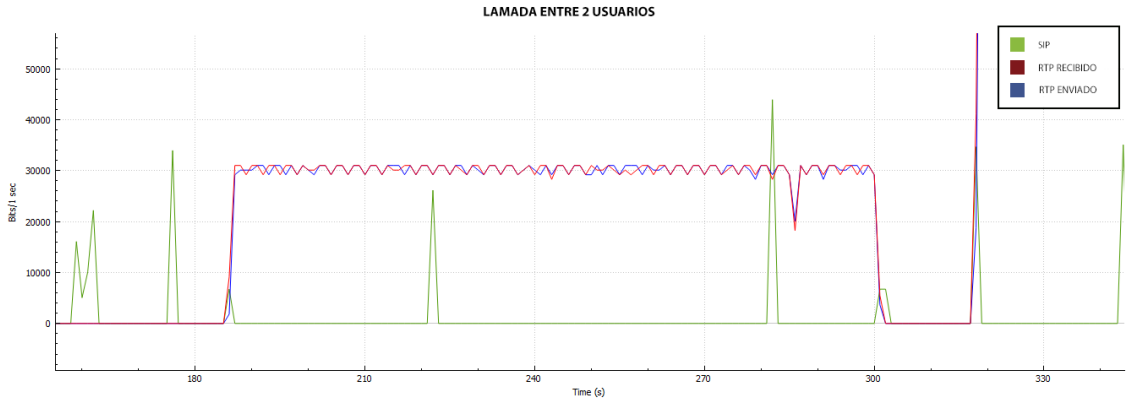

Fig. 32 Consumo de ancho de banda en una llamada

Tomando como ejemplo cualquiera de los correspondientes mensajes RTP, es posible comprobar que, efectivamente, el códec utilizado en el *payload* es el que se ha acordado previamente durante el establecimiento de sesión, en el que se especifica que será el G729; contiene un número de secuencia y el identificador de la fuente para poder reordenarse correctamente en el destino. Con los paquetes RTCP podemos hacer algo parecido, observando que el tipo de mensaje es *Sender Report*, por lo que informa tanto de lo enviado como de lo recibido.

```
> Frame 203: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface 0
> Ethernet II, Src: ZyxelCom_b7:87:5f (00:a0:c5:b7:87:5f), Dst: ZyxelCom_b7:88:c0 (00:a0:c5:b7:88:c0)
> Internet Protocol Version 4, Src: 172.1.2.17, Dst: 172.1.2.8
> User Datagram Protocol, Src Port: 2070, Dst Port: 2070
v Real-Time Transport Protocol
  > [Stream setup by SDP (frame 118)]
    10.. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0000 = Contributing source identifiers count: 0
    0... .... = Marker: False
    Payload type: ITU-T G.729 (18)
    Sequence number: 16
    [Extended sequence number: 65552]
    Timestamp: 8520
    Synchronization Source identifier: 0x3c2751e7 (1009209831)
    Payload: 79e6bcf9c42a14af53fc7daedef6a32a11533f10f9f90e50...
```

Fig. 33 Mensaje RTP

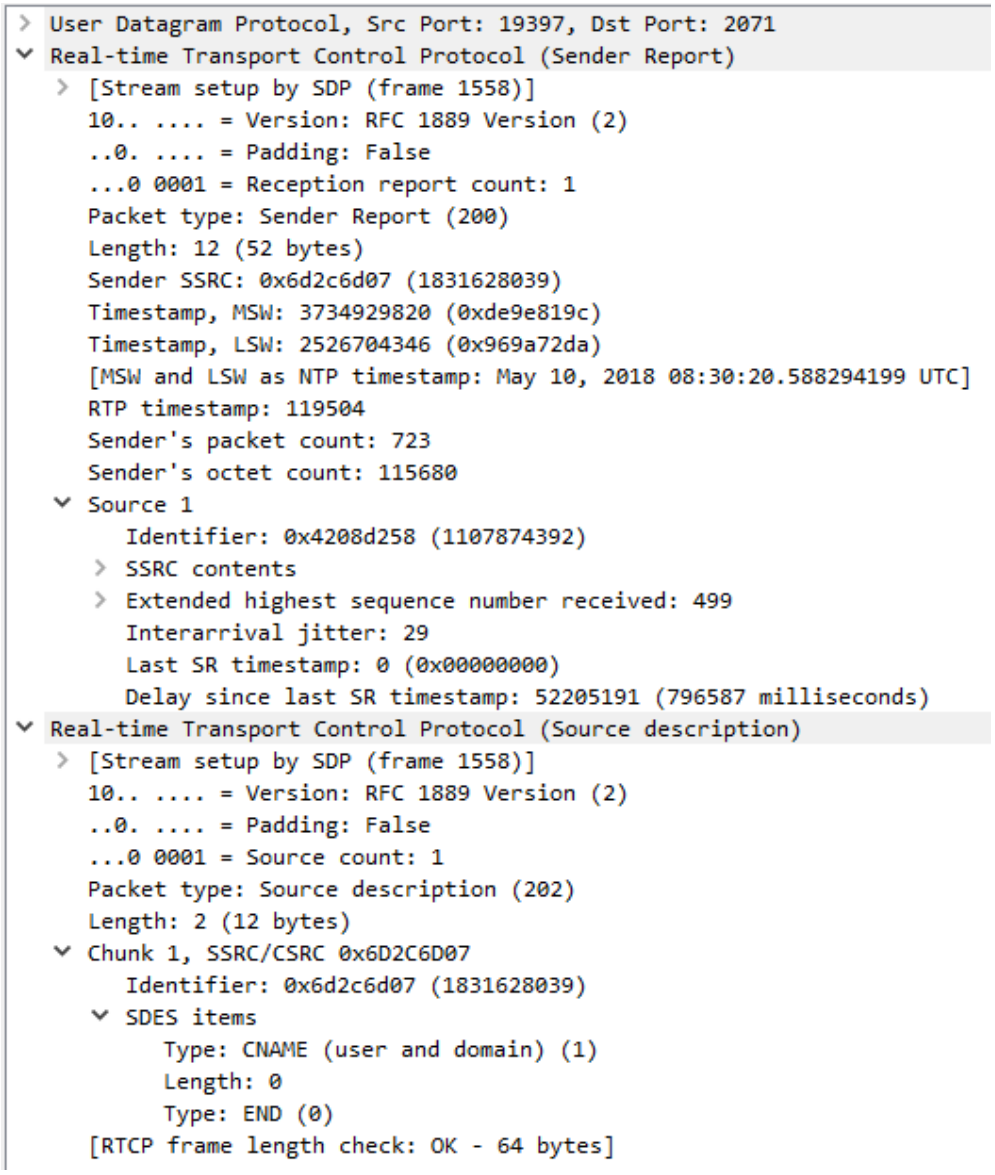

Fig. 34 Mensaje RTCP

4.3.2. BUSY

Como ya se ha comentado, la llamada puede ser rechazada por el receptor. Este envía una respuesta *486 BUSY* a la que el servidor responde con un *ACK*, y reenvía el mensaje al emisor que responderá con otro *ACK*.

Realizando otra prueba se puede comprobar que, efectivamente, esto es lo que ocurre. El establecimiento de llamada es igual que en el caso anterior (Fig. 26).

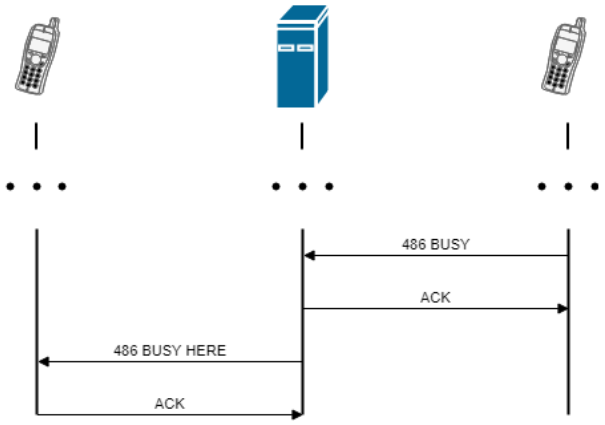

Fig. 35 Flujo de mensajes en una respuesta *BUSY*

4.3.3. CANCEL

El tercer caso es en el que el emisor es el que anula la llamada antes de que el receptor pueda responder aceptándola o rechazándola. El servidor responde con un mensaje tipo *2xx* indicando que se cancela la llamada con éxito. Luego reenvía esta solicitud al receptor y este responde de forma similar, indicando que se ha recibido. A continuación, se para el *ringing* y vuelve al estado original, que se anuncia con el mensaje *487 REQUEST TERMINATED.*

En una tercera prueba, y continuando también desde el punto en el que se intenta establecer la llamada (Fig. 26) se puede ver que el primer terminal envía el CANCEL con la URI del receptor, y recibe la confirmación de haberse cancelado una vez que el propio receptor acepta ese mensaje de cancelación.

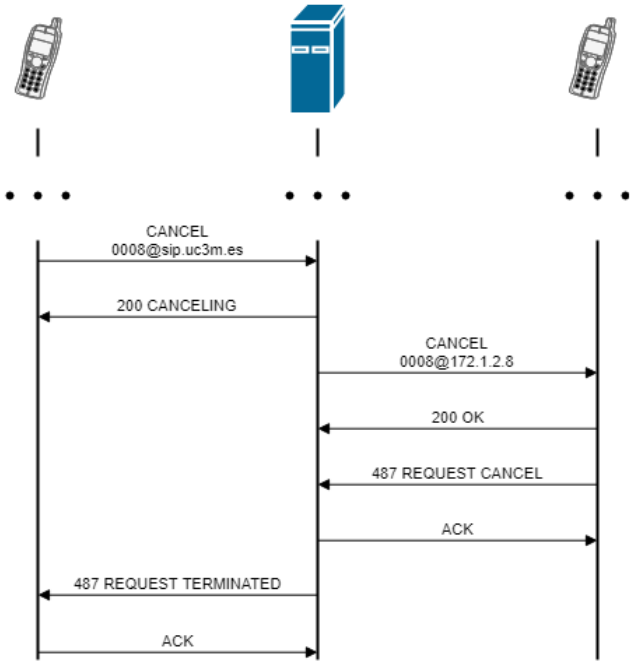

Fig. 36 Flujo de mensajes en una solicitud *CANCEL*

4.4. Conferencias

En cuanto a las conferencias, los usuarios deberán ser los que las creen. Para ello se marca en el terminal correspondiente el identificador de la conferencia que se quiera crear. Como hemos visto en el apartado de la configuración de Asterisk, este número tendrá que estar comprendido entre 8000 y 8999. Al marcarlo, el terminal envía un mensaje tipo *INVITE* con la URI-SIP de ese número, que recibirá Kamailio y procesará Asterisk.

```
241 12.28796… 172.1.2.20   172.1.2.99 SIP/SDP   1039 Request: INVITE sip:8762@sip.uc3m.es |
242 12.28827… 172.1.2.99   172.1.2.20 SIP        553 Status: 407 Proxy Authentication Required |
243 12.29068… 172.1.2.20   172.1.2.99 SIP/SDP   1039 Request: INVITE sip:8762@sip.uc3m.es |
244 12.29098… 172.1.2.99   172.1.2.20 SIP        553 Status: 407 Proxy Authentication Required |
245 12.29207… 172.1.2.99   172.1.2.20 SIP        553 Status: 407 Proxy Authentication Required |
246 12.29319… 172.1.2.99   172.1.2.20 SIP        553 Status: 407 Proxy Authentication Required |
247 12.32283… 172.1.2.20   172.1.2.99 SIP        446 Request: ACK sip:8762@sip.uc3m.es |
248 12.32461… 172.1.2.20   172.1.2.99 SIP/SDP   1220 Request: INVITE sip:8762@sip.uc3m.es |
249 12.32526… 172.1.2.20   172.1.2.99 SIP        446 Request: ACK sip:8762@sip.uc3m.es |
250 12.32570… 172.1.2.99   172.1.2.20 SIP        430 Status: 100 trying -- your call is important to us |
251 12.32598… 172.1.2.20   172.1.2.99 SIP/SDP   1220 Request: INVITE sip:8762@sip.uc3m.es |
252 12.32614… 172.1.2.99   172.1.2.20 SIP        430 Status: 100 trying -- your call is important to us |
253 12.33080… 172.1.2.99   172.1.2.20 SIP        430 Status: 100 trying -- your call is important to us |
254 12.33136… 172.1.2.99   172.1.2.20 SIP        430 Status: 100 trying -- your call is important to us |
255 12.33631… 172.1.2.99   172.1.2.20 SIP/SDP    971 Status: 200 OK |
256 12.33725… 172.1.2.20   172.1.2.99 SIP        446 Request: ACK sip:8762@sip.uc3m.es |
257 12.33780… 172.1.2.20   172.1.2.99 SIP        446 Request: ACK sip:8762@sip.uc3m.es |
```

Fig. 37 Intercambio de mensajes en creación de una conferencia

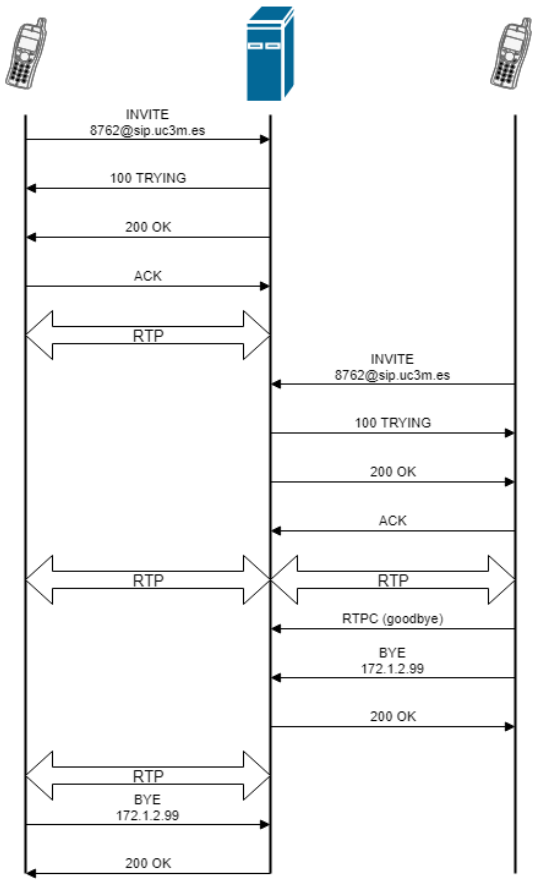

Fig. 38 Diagrama de creación de una conferencia

Tanto el foco como el *mixer* se encuentran en el mismo servidor ya que es la misma instancia de Asterisk la que lo gestiona. Automáticamente el creador es añadido a la conferencia que ha creado y ya empiezan a transmitirse mensajes RTP entre el servidor y este usuario. En cualquier momento desde que se crea la conferencia se puede unir un participante introduciendo el mismo identificador, que será enviado a través de un *INVITE* con esa URI. En la respuesta, sin embargo, no se genera ningún *ringing*, sino que la llamada es aceptada directamente tras un mensaje *trying* y un *OK*.

Desde la terminal de Asterisk se pueden comprobar todas las conferencias existentes en el momento mediante el comando "confbridge list", la lista de usuarios conectados usando "sip show peers", así como las llamadas totales realizadas con "core show calls".

```
Conference Bridge Name           Users  Marked Locked?
================================ ====== ====== ========
8473                                  1      1 unlocked
8139                                  1      1 unlocked
```

Fig. 39 Comando "confbridge list" en Asterisk CLI

```
Name/username             Host                                    Dyn Forcerport
 Comedia    ACL Port     Status      Description                     Realtime
0008/0008                  172.1.2.99                               D  Auto (No)
 No             5060     OK (87 ms)                                  Cached RT
0017/0017                  172.1.2.99                               D  Auto (No)
 No             5060     OK (70 ms)                                  Cached RT
2 sip peers [Monitored: 2 online, 0 offline Unmonitored: 0 online, 0 offline]
```

Fig. 40 Comando "sip show peers" en Asterisk CLI

```
2 active calls
4 calls processed
```

Fig. 41 Comando "core show calls" en Asterisk CLI

Hay que tener en cuenta que cada solicitud *INVITE* cuenta como una llamada, por tanto, cuando un usuario pretenda unirse a una conferencia, el número mínimo de llamadas procesadas será dos, una para crearla y otra para que un usuario se una, y cada vez que se una otro, aumentará en uno.

A diferencia de las llamadas normales, en las conferencias el que tiene que hacer la negociación es el servidor, y cada vez que un nuevo participante se una se volverá a realizar entre el servidor y el nuevo cliente.

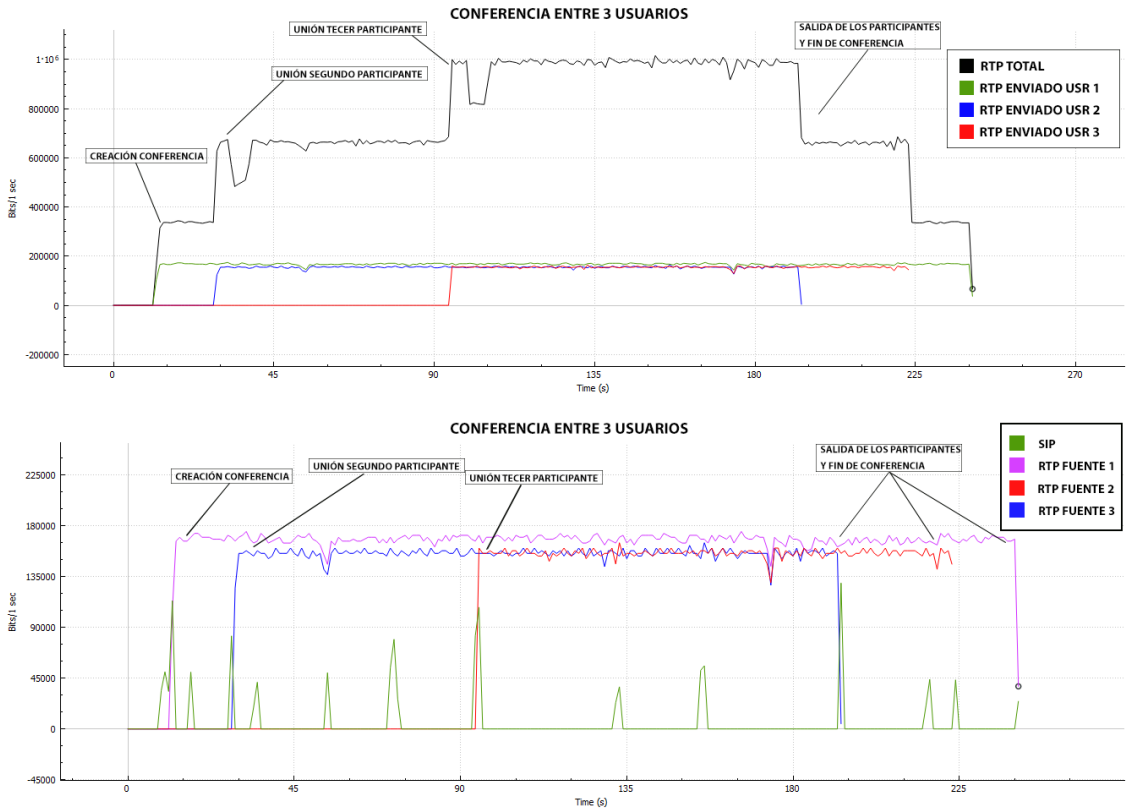

Fig. 42 Consumo de ancho de banda en una conferencia entre tres usuarios

En el caso de las conferencias, la comunicación RTP se realiza utilizando el códec de audio G.711, que tiene una menor compresión, mejor calidad de audio y, por tanto, mayor consumo.

5. PLANIFICACIÓN Y PRESUPUESTO

En este capítulo se indican los aspectos económicos relativos a los medios, planificación y realización de este. En la duración asociada a cada una de las actividades se incluye el tiempo dedicado al desarrollo de la propia actividad y la elaboración de la memoria que se presenta.

5.1. Definición de objetivos

El primer paso consiste en analizar y especificar los requisitos previos a la elaboración del proyecto. Para ello es necesario conocer las herramientas y protocolos empleados, así como la regulación vigente para su utilización dentro del marco europeo. Esto ha requerido la familiarización con la red interna de la Universidad Carlos III de Madrid, *router* Linksys wrt54gs y móviles VoIP *Wi-Fi* disponibles en la universidad, y establecer unas bases para el uso de sistemas de DNS, protocolos SIP, SDP, RTP. Duración: 2 semanas.

5.2. Planificación del proyecto

Una vez conocidos los elementos básicos que deben formar parte del trabajo se ha procedido a la descripción un sistema que integre los integre todos. De esta forma se estudian en profundidad cada uno de los elementos, de forma que se puedan integra todos en una sola estructura que funcione correctamente, llegando a la solución aplicada. Se estructura en tres partes:

- Estudio de DNS: con esto se busca que el sistema asocie nombres de dominio y direcciones IP. Se determina que el software empleado para esta tarea será BIND. Duración: 1 semana.
- Estudio de SIP y SDP: para conseguir que entre los diferentes terminales empleados exista una vía de comunicación en la que se generan, reciben, procesan y responden peticiones. Se concluye que se empleará Kamailio para el intercambio de estas peticiones. Duración: 2,5 semanas.
- Estudio de RTP: finalmente, con la consecución de una primera puesta en contacto entre terminales, se deben poder enviar datos multimedia entre ellos, ya sean solamente dos o más terminales. Se elige Asterisk para realizar este proceso. Duración: 2 semanas.

Duración total: 5,5 semanas.

5.3. Medios empleados

En esta sección se presenta una descripción de los materiales empleados, tanto hardware como software mencionados anteriormente:

Hardware:

- Ordenador portátil Lenovo con procesador Intel Core i5 con un sistema operativo Ubuntu 16.04 en el que se integra el servidor.
- *Router* Linksys wrt54gs para proporcionar conectividad inalámbrica entre los distintos componentes.
- Móviles VoIP *Wi-Fi* ZyXEL modelo p-2000w_v2 con los que se actuará como clientes.

Software:

- BIND *software open source* para realizar las tareas de asociación de nombres de dominio.
- Kamailio *software open source* que actuará como servidor *proxy*/registrador en el sistema, siendo por tanto responsable de SIP/SDP.
- Asterisk *software open source* con el que se procesarán las sesiones establecidas entre los terminales, ya sean llamadas o conferencias.
- MySQL como base de datos *open source* para el almacenamiento de la información relativa a los usuarios, que será compartida por Asterisk y Kamailio.

5.4. Desarrollo del trabajo

Tras establecer los medios que se emplean en la estructura se procede a la instalación y configuración de cada uno de los elementos. Tanto el sistema operativo Ubuntu como la base de datos se encontraban ya instalados en el ordenador utilizado, por lo que no han requerido mayor configuración; sin embargo, las tablas empleadas en la base de datos han sido añadidas después y el tiempo empleado para su instalación se computa en el de la instalación de Kamailio.

- BIND: instalación y puesta en marcha del sistema DNS. Duración: 1,5 semanas.
- Kamailio: instalación y configuración de un servidor *proxy*/registrador para generar un entorno de comunicación VoIP. Duración: 2 semanas.
- Asterisk: instalación, configuración e integración con Kamailio, de manera que se consiga un entorno de comunicación VoIP completo. Duración: 3,5 semanas.

Duración total: 7 semanas.

5.5. Pruebas

Por último, ha sido necesaria la elaboración de una batería de pruebas para comprobar que todos los elementos del sistema funcionaran correctamente, permitiendo la comunicación entre terminales y de acuerdo con los estándares actuales. Duración: 3 semanas.

5.6. Análisis económico

A continuación, se presenta el desglose de los costes asociados a la realización del proyecto. El tiempo y valor asociados a cada actividad es aproximado, realizándose entre los meses de febrero y junio de manera no consecutiva, en algunos casos tanto durante días laborales como fines de semana. No se incluyen gastos derivados de las herramientas y recursos utilizados, ya que estos son *software open source* o eran componentes que se encontraban ya disponibles al inicio del proyecto, como puede ser el ordenador portátil o los móviles. El valor del coste por hora se estima del salario medio de un ingeniero de telecomunicaciones de en entre 18 y 24 años, que resulta en aproximadamente 2000€ mensuales [23].

TABLA 3 PRESUPUESTO DEL PROYECTO

Tarea	Recursos empleados	Duración	Coste
Definición de objetivos	• Ordenador portátil Lenovo	2 semanas/6 horas diarias	20€/hora
Planificación del proyecto	• Ordenador portátil Lenovo	5,5 semanas/6 horas diarias	20€/hora
Desarrollo del trabajo	• Ordenador portátil Lenovo • Móviles VoIP • *Router* Linksys	7 semanas/6 horas diarias	20€/hora
Pruebas	• Ordenador portátil Lenovo • Móviles VoIP • *Router* Linksys • *Softphones*	3 semanas/6 horas diarias	20€/hora
	Total	17,5 semanas/600 horas	12000€

Se observa que la realización total del proyecto se ha completado a lo largo de 5 meses en un total de 600 horas, y que asciende a un coste total de 12000€.

6. CONCLUSIONES

6.1. Conclusión

El proyecto ha consistido en la creación de un entorno de comunicación de VoIP para prácticas docentes que permitiera, junto a un servicio de DNS, la comunicación de voz entre terminales basados en SIP. En el comienzo del proyecto se plantearon principalmente dos objetivos, por un lado, se pretendía ahondar en el conocimiento de la comunicación VoIP en unas condiciones verosímiles y, por otro, que pudieran servir para aplicarse a prácticas docentes.

Se ha establecido un servidor que permite que distintos usuarios puedan conectarse y comunicarse entre sí, cuyo control y descripción depende enteramente de quién lo lleve a cabo con lo que, una vez se haya aplicado a una clase práctica, los alumnos tendrán la posibilidad de comprobar el funcionamiento de las llamadas, conferencias y sistemas de DNS aplicados a VoIP ya que, desde el lado del servidor, se puede acceder a toda la información que es intercambiada, llamadas procesadas, usuarios activos, reconexión de usuarios, etc. Esto ayudará a entender la teoría y estándares explicados en clase mediante un uso práctico en el que elementos, como el propio teléfono móvil, pueden añadirse.

El cliente, por su parte, es capaz de realizar cualquiera de las funciones propias de un contexto de comunicación telefónica, tales como el registro o las llamadas. Esto puede hacerse sin necesidad de conocer el funcionamiento interno del sistema, tan solo con una IP, o un nombre de dominio y un puerto.

A nivel personal este proyecto ha servido para afianzar el conocimiento que poseía previamente de los sistemas de DNS y los protocolos relacionados con SIP implicados en la comunicación VoIP. Además, a medida que avanzaba el proyecto han ido surgiendo nuevos retos que no se habían planteado en un principio, y cuya aparición ha servido para aprender nuevos conceptos y plantear distintos enfoques de las situaciones.

Una vez finalizado el proyecto es posible decir que se han cumplido todos los requerimientos establecidos al comienzo, lo que a su vez permite que en el futuro se pueda desarrollar alguna otra actividad basada en lo expuesto anteriormente. Por todo esto se pueden considerar cumplidos los objetivos.

6.2. Líneas futuras

Desde este punto se pueden introducir una serie de mejoras que permitan desarrollar nuevas funcionalidades en el sistema ya establecido. Entre estas opciones están las siguientes:

- Buzón de voz: una de las tablas presentes en SQL es “voicemail”. Es posible configurar un entorno en el que, en caso de no establecerse una llamada tras una solicitud *INVITE*, el iniciador de la llamada pueda guardar un mensaje de audio

al destinatario, de manera que este destinatario sea capaz de acceder a dicho mensaje llamando a un número determinado.

- Videollamadas: entre las opciones a la hora de negociar el establecimiento de la llamada está el intercambio de vídeo en tiempo real. Sin embargo, su uso no ha sido posible en el laboratorio debido a las limitaciones propias de los dispositivos utilizados.
- Raspberry Pi: existe la posibilidad de instalar todo el servicio en este tipo de dispositivo de forma que permita su fácil conexión y desplazamiento, en lugar de iniciar todo de manera individual cada vez que se reinicia el sistema. De esta forma, el entorno podría estar funcionando con tan solo conectar dicho dispositivo.
- Distribución de carga: una de las ventajas que ofrece Kamailio es la posibilidad de repartir de manera equilibrada el trabajo entre varias instancias de Asterisk. Esto permite que el número de peticiones soportada aumente drásticamente y pueda ser utilizado en un entorno más amplio, algo que puede ser interesante explorar en una clase práctica.
- DHCP: *Dynamic Host Configuration Protocol*, Protocolo de Configuración Dinámica del Servidor en español, es un protocolo que permite la asignación automática de una serie de direcciones IP disponibles a los clientes. Tampoco fue posible implementarlo en la práctica debido a las limitaciones, el *router* disponible se encuentra configurado de forma que la asignación de IPs sea fija.

BIBLIOGRAFÍA

[1] Comisión Nacional de los Mercados y la Competencia, «Informe Económico Sectorial de las Telecomunicaciones y el Audiovisual,» 2017.

[2] ETSI, «European Telecommunications Standards Institute,» [En línea]. Available: http://www.etsi.org/.

[3] P. Mockapetris, "DOMAIN NAMES - CONCEPTS and FACILITIES", RFC 882, Noviembre 1983.

[4] P. Mockapetris, "DOMAIN NAMES - CONCEPTS and FACILITIES", RFC 883, Noviembre 1983.

[5] Gulbrandsen, A., Vixie, P. y L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, Febrero 2000.

[6] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. y E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, Junio 2002.

[7] Rosenberg, J. y H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, Junio 2002.

[8] Rosenberg, J. y H. Schulzrinne, "An Offer/Answer Model with the Session Description Protocol (SDP)", RFC 3264, Junio 2002.

[9] A. B. Roach, "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, Junio 2002.

[10] J. Rosenberg, "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, Febrero 2006.

[11] Johnston, A. y O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents", RFC 4579, Agosto 2006.

[12] Handley, M. y V. Jacobson, "SDP: Session Description Protocol", RFC 2327, Abril 1998.

[13] Schulzrinne, H., Casner, S., Frederick, R. y V. Jacobson, " RTP: A Transport Protocol for Real-Time Applications", RFC 3550, Julio 2003.

[14] Z. C. Corporation, «P-2000W_V2 User's Guide». 2005.

[15] Digium, Inc., «Asterisk,» [En línea]. Available: https://www.asterisk.org/.

[16] D.-C. Mierla, «Kamailio 4.0.x and Asterisk 11.3.0 Realtime Integration using Asterisk Database,» [En línea]. Available: http://kb.asipto.com/asterisk:realtime:kamailio-4.0.x-asterisk-11.3.0-astdb.

[17] The Kamailio SIP Server Project, «Kamailio,» [En línea]. Available: https://www.kamailio.org/w/.

[18] Digium, Inc., «Asterisk Project,» [En línea]. Available: https://wiki.asterisk.org/wiki/display/AST/Home.

[19] Universidad Carlos III de Madrid, «Kamailio: Manual de configuración y pruebas,» [En línea]. Available: https://www.etl.it.uc3m.es/Kamailio:_Manual_de_configuraci%C3%B3n_y_pruebas#Pruebas_con_los_tel.C3.A9fonos.

[20] Internet Systems Consortium, «BIND Open Source DNS Server,» [En línea]. Available: https://www.isc.org/downloads/bind/.

[21] D. Both, «Introduction to the Domain Name System (DNS),» [En línea]. Available: https://opensource.com/article/17/4/introduction-domain-name-system-dns.

[22] M. Anicas, «How To Configure BIND as a Private Network DNS Server on Ubuntu 14.04,» [En línea]. Available: https://www.digitalocean.com/community/tutorials/how-to-configure-bind-as-a-private-network-dns-server-on-ubuntu-14-04.

[23] Colegio Oficial Ingenieros de Telecomunicación, «Perfil del Ingeniero de Telecomunicación» 2017.

ANEXO A. FORMATO MENSAJES DNS

Todos los mensajes están divididos en cinco partes, cuyos valores de *ANSWER*, *AUTHORITY* Y *ADDITIONAL* pueden estar vacíos en ciertos casos:

HEADER
QUESTION
ANSWER
AUTHORITY
ADDITIONAL

HEADER:

La estructura de la cabecera es la siguiente:

TABLA 4. ESTRUCTURA DEL HEADER EN DNS

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ID															
QR	OPCODE				AA	TC	RD	RA		RCODE					
QDCOUNT															
ANCOUNT															
NSCOUNT															
ARCOUNT															

Fuente: RFC 883

Identifier (*ID*): 16 bits

Identificador generado por el programa que hace la consulta.

Query/Response (*QR)*: 1 bit

Especifica si el mensaje es consulta (0) o respuesta (1).

Operation Code (*OPCODE*): 4 bits

Define el tipo de respuesta según los valores:

- 0: estándar.
- 1: inversa.
- 2: solicitud completa que permite varias respuestas.
- 3: solicitud completa que permite una sola respuesta.
- 4-15: para futuro uso.

Authoritative Answer (*AA*, Respuesta Autoritaria): 1 bit

En las respuestas indica si el servidor que responde tiene autoridad sobre el dominio.

TrunCation (*TC*, TrunCamiento): 1 bit

Especifica si el mensaje está incompleto por exceder la longitud máxima de 512 caracteres.

Recursion Desired (*RD*, Deseo de Recursión): 1 bit

En una solicitud pide al servidor que la trate de manera recursiva. El soporte de este campo es opcional.

Recursion Available (*RA*, Disponibilidad de Recursión): 1 bit

Aclara si el servidor soporta la recursión.

Response Code (*RCODE*, Código de Respuesta): 4 bits

En las respuestas describe una serie de condiciones:

- 0: No hay error.
- 1: Error en el formato.
- 2: Fallo del servidor.
- 3: Error en el nombre de dominio de la solicitud.
- 4: Tipo de solicitud no implementado.
- 5: Operación rechazada.
- 6-15: para futuro uso.

QDCOUNT: 16 bits

Número de entradas en la sección de consulta.

ANCOUNT: 16 bits

Número de *RR*s en la sección de respuesta.

NSCOUNT: 16 bits

Número de entradas NS en la sección de registros de autoridad.

ASCOUNT: 16 bits

Número de RRs en la sección adicional.

QUESTION:

La sección de pregunta se usa en todas las peticiones salvo las inversas, que actuará como respuesta:

TABLA 5. ESTRUCTURA DE *QUESTION* EN DNS

0 1 2 3 4 5 6 7 8 9	1 0 1 1 1 2 1 3 1 4 1 5
QNAME . . .	
QTYPE	
QCLASS	
Fuente: RFC 883	

QNAME (Nombre): n octetos

Nombre reducido del dominio.

QTYPE (Tipo*)*: 16 bits

Contiene uno de los códigos que especifica el tipo de solicitud.

QCLASS (Clase): 16 bits

Define la clase de solicitud.

ANSWER, AUTHORITY y *ADDITIONAL*:

Las tres secciones (respuesta, autoridad y adicional) siguen el mismo patrón:

TABLA 6. ESTRUCTURA DE *ANSWER* EN DNS

0 1 2 3 4 5 6 7 8 9	1 0 1 1 1 2 1 3 1 4 1 5
NAME . . .	
TYPE	
CLASS	
TTL	
RDLENGTH	
RDATA . . .	
Fuente: RFC 883	

NAME (Nombre): n octetos

Nombre reducido del dominio al que pertenece este RR.

TYPE (Tipo*)*: 16 bits

Contiene uno de los códigos de RR y especifica el valor del campo *RDATA*.

CLASS (Clase): 16 bits

Define la clase de datos de *RDATA*.

Time To Live (TTL, Tiempo De Vida): 16 bits

Tiempo que el RR debe guardarse en *cache*.

RDLENGTH (Longitud de *RDATA*): 16 bits

Longitud que tiene el campo *RDATA* en octetos.

RDATA (Datos del Recurso): *rdlength* octetos

Cadena de octetos que describe el recurso.

ANEXO B. DESCRIPTORES SDP

Version (Versión)

Siempre será “v=0”.

Origin (Origen)

Identifica al creador y actúa como identificador global único de la sesión.

o=<username><session id><version><network type><address type><address>

Ej: o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4

- *username*: identificador del usuario que originó la sesión (no puede contener espacios).
- *session id*: cadena de caracteres numéricos que identifican a la sesión.
- *version*: indica la versión del anuncio en caso de que se hayan enviado varios mensajes durante la sesión. Así, el *proxy* utilizará el más reciente.
- *network type*: tipo de red, al igual que en DNS será “IN”.
- *address type*: según sea IPv4 o IPv6 utilizará IP4 o IP6 respectivamente.
- *address*: dirección IP del creador

Session name (Nombre de la sesión)

Solamente puede haber uno por descriptor de sesión.

s=<session name>

Ej: s=SDP Seminar

Session and Media Information (Información de la sesión y multimedia)

Solamente puede haber uno por descriptor de sesión o de multimedia. Es opcional

i=<descripción de sesión>

Ej: i=A Seminar on the session description protocol

URI

Contiene información adicional sobre la conferencia. Solamente puede haber uno por descriptor de sesión. Es opcional.

u=<URI>

Ej: u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps

Correo electrónico y número de teléfono

Proporcionan información de contacto del responsable, que no tiene por qué ser el creador. Solamente puede haber uno por descriptor de sesión. Son opcionales, aunque uno de los dos debe estar presente. Pueden tomar varios valores y contener una cadena de caracteres que suele ser el nombre de la persona a contactar.

e=<dirección de correo>

Ej: e=mjh@isi.edu (Mark Handley)

p=<número de teléfono>

Ej: p=+44-171-380-7777

Connection Data (Información de la conexión)

Puede haber un campo "c=" en cada descripción multimedia o uno a nivel de sesión.

c=<network type> <address type> <connection address>

Ej: c=IN IP4 224.2.1.1/127

- *network type*: tipo de red que identifica (IN).
- *address type*: según sea IPv4 o IPv6 utilizará IP4 o IP6 respectivamente.
- *connection address*: dirección de conexión. Puede especificarse el TTL precedido de una "/".

Bandwidth (Ancho de Banda)

Especifica el ancho de banda que deberá usarse en la sesión. Es opcional.

b=<modifier>:<bandwidth-value>

- *modifier*: una palabra aplica un modificador al ancho de banda.
  - Conferencia Total (CT): asocia un máximo ancho de banda y da una idea de si dos conferencias pueden coexistir.
  - Application-Specific máximum (AS): el ancho de banda máximo es interpretado por cada aplicación.
- *bandwidth-value*: valor dado en kilobits por segundo.

Times, Repeat Times, Time Zones (Tiempos, Tiempos de Repetición, Zonas Horarias)

Expresado en segundos, aunque admite "d" (días), "h" (horas), "m" (minutos) como tiempo equivalente.

t=<start time> <stop time>

r=<repeat interval> <active duration> <list of offsets from start-time>

z=<adjustment time> <offset> <adjustment time> <offset> . . .

- "t=" indica los tiempos de inicio y finalización de una sesión. Se pueden utilizar varios si la sesión está activa de forma irregular, o complementarlo con "r=" si es regular.
- "r=" indica el intervalo de tiempo en que debe repetirse. Expresado en segundos
  - *repeat-interval*: cada cuánto tiempo se repite la sesión.
  - *active duration*: tiempo que permanece activa.
  - *offsets from start-time*: tiempo de espera desde el inicio de la sesión hasta que se activa.
- "z=" indica la zona horaria en la que se ha creado la sesión y permite adaptarla, por ejemplo, para el cambio horario de verano-invierno.
  - *adjustment time*: tiempo en el que debe realizarse el ajuste.
  - *offset*: diferencia horaria.

*Encryption Key*s (Claves de Cifrado)

Es opcional, pero si existe debe ir antes que la primera entrada multimedia.

k=<method>:<encryption key>

- *method*: indica el mecanismo para obtener la clave. Puede ser una URI, un código base64, etc.
- *encryption key*: sobre este campo se aplica el método de obtención de clave.

Attributes (Atributos)

Pueden ser a nivel de sesión, multimedia o ambos.

a=<attribute>:<value>

Ej: a=rtpmap:98 L16/11025/2

Hay dos clases: propiedad (propio de una sesión) y de valor (contiene un valor adicional).

Media Announcements (Declaraciones Multimedia)

Describe la información multimedia. El campo "m=" termina donde empieza el siguiente o si acaba la descripción multimedia.

m=<media> <port> <transport> <fmt list>

Ej: m=audio 49230 RTP/AVP 96 97 98

a=rtpmap:96 L8/8000

a=rtpmap:97 L16/8000

a=rtpmap:98 L16/11025/2

- *media*: tipo de datos multimedia transmitidos (“audio”, “video”, “application”, “data”, “control”).
- *port*: puerto al que se envían esos datos.
- *transport*: protocolo de transporte usado, que irá sobre el protocolo especificado en “c=”. Puede ser RTP/AVP (audio y vídeo) o UDP.
- *fmt list*: lista de formatos soportados. Se definen con más detalle usando el campo “a=”.

ANEXO C. FORMATO RTP Y RTCP

RTP

TABLA 7. ESTRUCTURA DE MENSAJES RTP

0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5	6 7 8 9 0 1 2 3	4 5 6 7 8 9 0 1
V	P X CC	M PT	NUMERO DE SECUENCIA
TIMESTAMP			
SSRC			
CSRC			
CSRC (2)			
Fuente: RFC 3550			

Cada fila tiene una longitud de 32 bits (4 octetos). Los 12 primeros octetos son comunes a todo paquete RTP, mientras que la lista de CSRCs aparece después de ser añadido por un *mixer*.

Version (V): 2 bits

Tiene el valor “2”.

Padding (P): 1 bit

Marca si existen octetos de relleno al final del *payload*. El último octeto de relleno establece el número de octetos que hay de relleno y deben ser ignorados.

Extension (X): 1 bit

Si está activado indica que la cabecera va seguida de una extensión.

CSRC Count (CC): 4 bits

Número de identificadores CSRC que siguen al resto de cabeceras.

Marker (M): 1 bit

Marca eventos relevantes relacionados con el paquete.

Payload Type (PT): 7 bits

Identifica el formato del *payload* y cómo debe actuar la aplicación receptora del paquete.

Sequence Number (Número de Secuencia): 16 bits

Empieza con un valor aleatorio y aumenta en uno para cada paquete RTP enviado. Se utiliza para detectar pérdidas.

Timestamp (Marca de Tiempo): 32 bits

> Marca el instante de muestreo del primer octeto del paquete RTP. El reloj interno debe tener suficiente precisión como para que pueda ser utilizado para una correcta sincronización de los paquetes. Su primer valor es aleatorio.

SSRC (*Synchronization Source*, Fuente de Sincronización): 32 bits

> Identificador único de una fuente en una sesión RTP. Se genera de forma aleatoria buscando que no existan dos fuentes con el mismo identificador mediante algoritmos concretos. En el receptor se agrupan todos los paquetes recibidos de un mismo SSRC en una sesión RTP para su reproducción.

CSRC list (*Contributing Sources List*, Lista de Fuentes Contribuyentes): 32 bits (cada una)

> Identifica hasta un máximo de 15 elementos (indicado por el campo "CC") que han sido añadidos por un *mixer*.

RTCP

Paquete SR

Dividido en tres secciones, seguida de una cuarta opcional.

TABLA 8. ESTRUCTURA DE PAQUETES *SENDER REPORT*

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
V	P	RC	PT=SR=200	LENGTH
SSRC SENDER			
NTP TIMESTAMP (most significant word)			
NTP TIMESTAMP (least significant word)			
RTP TIMESTAMP			
SENDER PACKET COUNT			
SENDER OCTECT COUNT			
SSRC_1			
FRACTION LOST	PACKET LOST CUMULATIVE		
EXTENDED HIGHEST SEQUENCE NUMBER RECEIVED			
INTERARRIVAL JITTER			
LSR			
DLSR			
. . .			

EXTENSIONS
Fuente: RFC 3550

Paquete RR

Solamente son dos partes más una opcional. El formato del RR es el mismo que el SR salvo que el PT es 201 y no se envía información del emisor.

TABLA 9. ESTRUCTURA DE PAQUETES *RECEIVER REPORT*

0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5 6 7 8 9 0 1
V | P	RC	PT=SR=201 | LENGTH
SSRC SENDER		
SSRC_1		
FRACTION LOST	PACKET LOST CUMULATIVE	
EXTENDED HIGHEST SEQUENCE NUMBER RECEIVED		
INTERARRIVAL JITTER		
LSR		
DLSR		
. . .		
EXTENSIONS		
Fuente: RFC 3550		

La primera consiste en 8 octetos

Version (V): 2 bits

Tiene el valor “2”.

Padding (P): 1 bit

Marca si existen octetos de relleno al final del *payload*. El último octeto de relleno establece el número de octetos que hay de relleno y deben ser ignorados.

Reception Report Count (RC): 5 bits

Número de bloques de recepción que contiene el paquete.

Packet Type (PT): 8 bits

Contiene el valor “20x” que identifica el tipo de paquete.

Length (Longitud): 16 bits

Longitud del paquete en palabras de 32 bits menos una.

SSRC Sender (*Synchronization Source Sender*): 32 bits

Identificador del emisor de este paquete.

La segunda sección aparece en todos los paquetes SR, es un resumen de la información de transmisión del emisor, y consta de 20 octetos:

NTP Timestamp (*Network Time Protocol Timestamp*, Marca de Tiempo de Protocolo de Tiempo de Red): 64 bits

Representa un tiempo absoluto en el que se envió el informe, expresado en los segundos transcurridos desde el 1 de enero de 1900.

RTP Timestamp (Marca de Tiempo de RTP): 32 bits

El mismo dato que *NTP Timestamp* pero con las misma unidades y *offset* que las marcas temporales de los paquetes RTP.

Sender's Packet Count (Contador de Paquetes del Emisor): 32 bits

Total de paquetes transmitidos por el emisor hasta la generación de este paquete SR. Se reinicia si el emisor cambia su SSRC.

Sender's Octet Count (Contador de Octetos del Emisor): 32 bits

Total de octetos del *payload* transmitidos por el emisor hasta la generación de este paquete SR. Se reinicia si el emisor cambia su SSRC.

La tercera sección está compuesta por cero o más bloques de reporte dependiendo del número de fuentes de las que ha recibido el emisor desde el último informe. Contiene estadísticas de la información recibida de cada SSRC y permite calcular el tiempo de ida y vuelta a las fuentes.

SSRC_n: 32 bits

Identificador de la fuente a la que hace referencia el bloque.

Fraction Lost (Fracción de Pérdidas): 8 bits

Fracción de paquetes RTP perdida desde el último SR o RR. Definida como el número de paquetes perdidos partido por el total de paquetes esperados.

Cumulative Number of Packets Lost (Número Acumulativo de Paquetes Perdidos): 24 bits

Total de paquetes perdidos desde la fuente SSRC_n desde el inicio de la recepción. Es el número de paquetes esperados menos el de recibidos.

Extended Highest Sequence Number (Mayor Número de Secuencia Extendido): 32 bits

Los primeros 16 bits indican el número de ciclos que ha habido. Los últimos 16 bits contienen el número de secuencia más alto recibido desde SSRC_n en un paquete.

Interarrival Jitter (*Jitter* entre llegadas): 32 bits

Variación de llegadas de los paquetes RTP.

LSR (*Last SR Timestamp*, Última Marca de Tiempo de SR): 32 bits

Contiene los 32 bits del medio del *NTP Timestamp* recibido en el último SR.

DLSR (*Delay Since Last SR*, Retardo Desde el Último SR): 32 bits

El retardo desde la recepción del último SR desde SSRC_n hasta el envío de este reporte.

Paquete SDES

TABLA 10. ESTRUCTURA DE PAQUETES *SDES*

0 1 2 3 4 5 6 7	8 9 0 1 2 3 4 5	6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
V	P RC	PT=SR=202 LENGTH
SSRC/CSRC_1		
SDES items		
SSRC/CSRC_2		
SDES items		

Fuente: RFC 3550

El paquete SDES es una estructura de tres niveles compuesta de una cabecera, similar a los paquetes SR y RR, y cero o más secciones que describen una fuente.

Cada sección consiste en un identificador SSRC/CSRC junto a una lista de elementos con información acerca de ese identificador. Cada elemento consta de un campo de 8 bits que indica el número de octetos del texto (descriptores). Solamente el CNAME es obligatorio.

Paquete BYE

Indica la razón por la que una o más fuentes dejan de estar activas.

TABLA 11. ESTRUCTURA DE PAQUETES *BYE*

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
V	P	SC	PT=SR=203	LENGTH
SSRC/CSRC			
. . .			
LENGTH		REASON FOR LEAVING	
Fuente: RFC 3550			

Si es recibido por un mixer, este lo reenviará sin cambiar los campos SSRC/CSRC. Si el *mixer* es el que envía el BYE incluirá todos los SSRC/CSRC que gestiona, así como el suyo propio. La razón del abandono es opcional. Si se añade tiene un campo de 8 bits que indica los octetos del texto que establece la razón para abandonar.

Paquete APP

Para uso experimental en aplicaciones y características en desarrollo.

TABLA 12. ESTRUCTURA DE PAQUETES *APP*

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31
V	P	SUBTYPE	PT=SR=204	LENGTH
SSRC/CSRC			
NAME (ASCII)			
APPLICATION-DEPENDENT DATA			
Fuente: RFC 3550			

- *Subtype*: permite definir un conjunto de paquetes de aplicaciones bajo un mismo nombre.
- *Name*: nombre elegido por el creador de los paquetes de aplicación para que sea único respecto a otros paquetes que la aplicación pueda recibir.
- *Application-Dependent Data* (Datos Dependientes de la Aplicación): son interpretados por la propia aplicación.

ANEXO D. INSTALACIÓN DE SOFTWARE

BIND

Podemos realizar todos los comandos como *root*:

```
$ sudo su -
```

Primero actualizamos la lista de paquetes:

```
$ apt-get update
```

Ahora instalamos BIND:

```
$ apt-get install bind9 bind9utils bind9-doc
```

Establecemos BIND como IPv4:

```
$ vi /etc/default/bind9
```

Y dentro del archivo definimos:

```
OPTIONS=”-4 -u bind”
```

Asterisk

Obtenemos privilegios de *root*:

```
$ sudo su -
```

Primero necesitaremos tener instalados un servidor MySQL y las librerías de UnixODBC:

```
$ apt-get install mysql-server
$ apt-get install libmysqlclient-dev
```

```
$ apt-get install unixodbc-dev

$ apt-get install libmyodbc
```

Descargamos, descomprimimos y configuramos la versión deseada:

```
$ cd /usr/local/src

$ wget http://downloads.asterisk.org/pub/telephony/asterisk/releases/asterisk-14.7.6.tar.gz

$ tar xvfz asterisk-14.7.6.tar.gz

$ cd asterisk-14.7.6

$ ./configure
```

Para activar algunas características como el idioma o prefijos telefónicos de cada país:

```
$ make menuselect
```

A continuación, compilamos e instalamos:

```
$ make

$ make install
```

Creamos una base de datos MySQL para guardar toda la información relativa a los usuarios:

```
CREATE DATABASE asterisk;

USE asterisk;

GRANT ALL ON asterisk.* TO asterisk@localhost IDENTIFIED BY 'asterisk_password';

DROP TABLE IF EXISTS sipusers;
CREATE TABLE `sipusers` (
    `id` INT(11) NOT NULL AUTO_INCREMENT,
     `name` VARCHAR(10) NOT NULL,
     `ipaddr` VARCHAR(15) DEFAULT NULL,
```

```
`port` INT(5) DEFAULT NULL,
`regseconds` INT(11) DEFAULT NULL,
`defaultuser` VARCHAR(10) DEFAULT NULL,
`fullcontact` VARCHAR(35) DEFAULT NULL,
`regserver` VARCHAR(20) DEFAULT NULL,
`useragent` VARCHAR(20) DEFAULT NULL,
`lastms` INT(11) DEFAULT NULL,
`host` VARCHAR(40) DEFAULT NULL,
`type` enum('friend','user','peer') DEFAULT NULL,
`context` VARCHAR(40) DEFAULT NULL,
`permit` VARCHAR(40) DEFAULT NULL,
`deny` VARCHAR(40) DEFAULT NULL,
`secret` VARCHAR(40) DEFAULT NULL,
`md5secret` VARCHAR(40) DEFAULT NULL,
`remotesecret` VARCHAR(40) DEFAULT NULL,
`transport` enum('udp','tcp','udp,tcp','tcp,udp') DEFAULT NULL,
`dtmfmode` enum('rfc2833','info','shortinfo','inband','auto') DEFAULT NULL,
`directmedia` enum('yes','no','nonat','update') DEFAULT NULL,
`nat` enum('yes','no','never','route') DEFAULT NULL,
`callgroup` VARCHAR(40) DEFAULT NULL,
`pickupgroup` VARCHAR(40) DEFAULT NULL,
`language` VARCHAR(40) DEFAULT NULL,
`disallow` VARCHAR(40) DEFAULT NULL,
`allow` VARCHAR(40) DEFAULT NULL,
`insecure` VARCHAR(40) DEFAULT NULL,
`trustrpid` enum('yes','no') DEFAULT NULL,
`progressinband` enum('yes','no','never') DEFAULT NULL,
`promiscredir` enum('yes','no') DEFAULT NULL,
`useclientcode` enum('yes','no') DEFAULT NULL,
`accountcode` VARCHAR(40) DEFAULT NULL,
`setvar` VARCHAR(40) DEFAULT NULL,
`callerid` VARCHAR(40) DEFAULT NULL,
`amaflags` VARCHAR(40) DEFAULT NULL,
`callcounter` enum('yes','no') DEFAULT NULL,
`busylevel` INT(11) DEFAULT NULL,
`allowoverlap` enum('yes','no') DEFAULT NULL,
`allowsubscribe` enum('yes','no') DEFAULT NULL,
`videosupport` enum('yes','no') DEFAULT NULL,
`maxcallbitrate` INT(11) DEFAULT NULL,
`rfc2833compensate` enum('yes','no') DEFAULT NULL,
`mailbox` VARCHAR(40) DEFAULT NULL,
`session-timers` enum('accept','refuse','originate') DEFAULT NULL,
`session-expires` INT(11) DEFAULT NULL,
`session-minse` INT(11) DEFAULT NULL,
`session-refresher` enum('uac','uas') DEFAULT NULL,
`t38pt_usertpsource` VARCHAR(40) DEFAULT NULL,
`regexten` VARCHAR(40) DEFAULT NULL,
`fromdomain` VARCHAR(40) DEFAULT NULL,
`fromuser` VARCHAR(40) DEFAULT NULL,
`qualify` VARCHAR(40) DEFAULT NULL,
```

```
    `defaultip` VARCHAR(40) DEFAULT NULL,
    `rtptimeout` INT(11) DEFAULT NULL,
    `rtpholdtimeout` INT(11) DEFAULT NULL,
    `sendrpid` enum('yes','no') DEFAULT NULL,
    `outboundproxy` VARCHAR(40) DEFAULT NULL,
    `callbackextension` VARCHAR(40) DEFAULT NULL,
    `timert1` INT(11) DEFAULT NULL,
    `timerb` INT(11) DEFAULT NULL,
    `qualifyfreq` INT(11) DEFAULT NULL,
    `constantssrc` enum('yes','no') DEFAULT NULL,
    `contactpermit` VARCHAR(40) DEFAULT NULL,
    `contactdeny` VARCHAR(40) DEFAULT NULL,
    `usereqphone` enum('yes','no') DEFAULT NULL,
    `textsupport` enum('yes','no') DEFAULT NULL,
    `faxdetect` enum('yes','no') DEFAULT NULL,
    `buggymwi` enum('yes','no') DEFAULT NULL,
    `auth` VARCHAR(40) DEFAULT NULL,
    `fullname` VARCHAR(40) DEFAULT NULL,
    `trunkname` VARCHAR(40) DEFAULT NULL,
    `cid_number` VARCHAR(40) DEFAULT NULL,
    `callingpres`
enum('allowed_not_screened','allowed_passed_screen','allowed_failed_screen'
                    ,'allowed','prohib_not_screened','prohib_passed_screen','prohib_failed_
screen'
                    ,'prohib') DEFAULT NULL,
    `mohinterpret` VARCHAR(40) DEFAULT NULL,
    `mohsuggest` VARCHAR(40) DEFAULT NULL,
    `parkinglot` VARCHAR(40) DEFAULT NULL,
    `hasvoicemail` enum('yes','no') DEFAULT NULL,
    `subscribemwi` enum('yes','no') DEFAULT NULL,
    `vmexten` VARCHAR(40) DEFAULT NULL,
    `autoframing` enum('yes','no') DEFAULT NULL,
    `rtpkeepalive` INT(11) DEFAULT NULL,
    `call-limit` INT(11) DEFAULT NULL,
    `g726nonstandard` enum('yes','no') DEFAULT NULL,
    `ignoresdpversion` enum('yes','no') DEFAULT NULL,
    `allowtransfer` enum('yes','no') DEFAULT NULL,
    `dynamic` enum('yes','no') DEFAULT NULL,
    `sippasswd` VARCHAR(80) DEFAULT NULL,
    PRIMARY KEY (`id`),
    UNIQUE KEY `name` (`name`),
    KEY `ipaddr` (`ipaddr`,`port`),
    KEY `host` (`host`,`port`)
) ENGINE=MyISAM;


DROP TABLE IF EXISTS sipregs;
CREATE TABLE `sipregs` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(80) NOT NULL DEFAULT '',
```

```
  `fullcontact` VARCHAR(80) NOT NULL DEFAULT '',
  `ipaddr` VARCHAR(45) DEFAULT NULL,
  `port` mediumint(5) UNSIGNED NOT NULL DEFAULT '0',
  `username` VARCHAR(80) NOT NULL DEFAULT '',
  `regserver` VARCHAR(100) DEFAULT NULL,
  `regseconds` INT(11) NOT NULL DEFAULT '0',
  `defaultuser` VARCHAR(80) NOT NULL DEFAULT '',
  `useragent` VARCHAR(20) DEFAULT NULL,
  `lastms` INT(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
);


DROP TABLE IF EXISTS voicemail;
CREATE TABLE voicemail (
        uniqueid INT(5) NOT NULL AUTO_INCREMENT PRIMARY KEY,
        context CHAR(80) NOT NULL DEFAULT 'default',
        mailbox CHAR(80) NOT NULL,
        password CHAR(80) NOT NULL,
        fullname CHAR(80),
        email CHAR(80),
        pager CHAR(80),
        attach CHAR(3),
        attachfmt CHAR(10),
        serveremail CHAR(80),
        LANGUAGE CHAR(20),
        tz CHAR(30),
        deletevoicemail CHAR(3),
        saycid CHAR(3),
        sendvoicemail CHAR(3),
        review CHAR(3),
        tempgreetwarn CHAR(3),
        operator CHAR(3),
        envelope CHAR(3),
        sayduration CHAR(3),
        saydurationm INT(3),
        forcename CHAR(3),
        forcegreetings CHAR(3),
        callback CHAR(80),
        dialout CHAR(80),
        exitcontext CHAR(80),
        maxmsg INT(5),
        volgain DECIMAL(5,2),
        imapuser VARCHAR(80),
        imappassword VARCHAR(80),
        imapsever VARCHAR(80),
        imapport VARCHAR(8),
        imapflags VARCHAR(80),
        stamp TIMESTAMP
```

```
);

DROP TABLE IF EXISTS voicemail_data;
CREATE TABLE voicemail_data (
	filename CHAR(255) NOT NULL PRIMARY KEY,
	origmailbox CHAR(80),
	context CHAR(80),
	macrocontext CHAR(80),
	exten CHAR(80),
	priority INT(5),
	callerchan CHAR(80),
	callerid CHAR(80),
	origdate CHAR(30),
	origtime INT(11),
	category CHAR(30),
	duration INT(11)
);
DROP TABLE IF EXISTS voicemail_messages;
CREATE TABLE voicemail_messages (
	dir CHAR(255),
	msgnum INT(4),
	context CHAR(80),
	macrocontext CHAR(80),
	callerid CHAR(80),
	origtime INT(11),
	duration INT(11),
	recording BLOB,
	flag CHAR(30),
	category CHAR(30),
	mailboxuser CHAR(30),
	mailboxcontext CHAR(30),
	msg_id CHAR(40),
	PRIMARY KEY (dir, msgnum)
);
```

El acceso a la base de datos de asterisk estará determinado por la contraseña "asterisk_password" de la tercera línea:

```
GRANT ALL ON asterisk.* TO asterisk@localhost IDENTIFIED BY
'asterisk_password';
```

La tabla "sipusers" contiene información relativa a los usuarios que se registren, como su id, nombre de usuario, contraseña, dirección IP, tipo de usuario, etc.

La tabla "sipregs" guarda los registros SIP e incluye datos como nombre de usuario, UA o la última distancia conocida hasta el UA en milisegundos.

En "voicemail" están los perfiles de los usuarios para acceder al correo de voz.

Las tablas "voicemail_data" y "voicemail_messages" recogen información específica de los mensajes de voz (origen, extensiones, el propio mensaje, etc).

Editamos el archivo “/etc/odbcinst.ini”:

```
$ vi /etc/odbcinst.ini
```

Y añadimos:

```
[MySQL]
Description = MySQL driver
Driver = libmyodbc.so
Setup = libodbcmyS.so
CPTimeout =
CPReuse =
UsageCount = 1
```

Hacemos lo mismo para “/etc/odbc.ini”, cambiando el valor de PASSWORD en caso de haberlo modificado en la base de datos:

```
[MySQL-asterisk]
Description = MySQL Asterisk database
Trace = Off
TraceFile = stderr
Driver = MySQL
SERVER = localhost
USER = asterisk
PASSWORD = asterisk_password
PORT = 3306
DATABASE = asterisk
```

El archivo “/etc/asterisk/res_odbc.conf” será el que nos de acceso a la base de datos:

```
[asterisk]
enabled => yes
dsn => MySQL-asterisk
username => asterisk
password => asterisk_password
pre-connect => yes
```

Y en “/etc/asterisk/extconfig.conf”:

```
sipusers => odbc,asterisk,sipusers
sippeers => odbc,asterisk,sipusers
sipregs => odbc,asterisk,sipregs
voicemail => odbc,asterisk,voicemail
```

Kamailio

Para la instalación necesitaremos acceso *root*:

```
$ sudo su -
```

Empezaremos por crear el directorio donde se guardarán las fuentes:

```
$ mkdir -p /usr/local/src/kamailio-4.4
$ cd /usr/local/src/kamailio-4.4
```

Descargamos las fuentes GIT:

```
$ git clone --depth 1 --no-single-branch https://github.com/kamailio/kamailio kamailio

$ cd kamailio

$ git checkout -b 4.4 origin/4.4
```

Generamos los archivos de configuración especificando los módulos extra que queramos añadir:

```
$ make include_modules=”db_mysql dialplan” cfg
```

Estos módulos se pueden cambiar directamente en la variable “include_modules” del archivo “modules.lst”.

Compilamos Kamailio:

```
$ make all
```

Y lo instalamos:

```
$ make install
```

Se instalará en la dirección “/usr/local/sbin” y el archivo de configuración que más tarde tendremos que cambiar estará en “/usr/local/etc/kamailio/kamailio.cfg”.

Crearemos a su vez una base de datos con la que se podrán administrar los registros de los clientes. Empezamos editando el archivo “kamctlrc”:

```
$ nano -w /usr/local/etc/kamailio/kamctlrc
```

Y establecemos el motor la base de datos a MySQL:

```
DBENGINE=MYSQL
```

Creamos la base de datos:

```
$ /usr/local/sbin/kamdbctl create
```

Esta base de datos tendrá el nombre “kamailio”, y los usuarios “kamailio” con contraseña “kamailiorw” y acceso total, y “kamailioro” con contraseña “kamailioro” y solamente acceso a lectura.

Editamos el archivo de configuración:

```
$ nano -w /usr/local/etc/kamailio/kamailio.cfg
```

Añadimos en la parte superior después de “#!KAMAILIO”:

```
#!define WITH_MYSQL
#!define WITH_AUTH
#!define WITH_USRLOCDB
```

Con Kamailio estará instalado correctamente. Para ejecutarlo copiamos el archivo de inicialización .init en “/etc/init.d/” y nos concedemos permisos sobre él

```
$ cp /usr/local/src/kamailio-4.4/kamailio/pkg/kamailio/deb/debian/kamailio.init
/etc/init.d/kamailio

$ chmod 755 /etc/init.d/kamailio
```

Modificamos este archivo:

```
$ nano -w /etc/init.d/kamailio
```

Actualizando los valores de $DAEMON y $CFGFILE:

```
DAEMON=/usr/local/sbin/kamailio
CFGFILE=/usr/local/etc/kamailio/kamailio.cfg
```

Igualmente copiamos el archivo de configuración .default en el directorio "/etc/default":

```
$ cp /usr/local/src/kamailio-4.4/kamailio/pkg/kamailio/deb/debian/kamailio.default
/etc/default/kamailio.default

$ chmod 755 /etc/default/kamailio.default
```

Lo abrimos:

```
$ nano -w /etc/default/kamailio.default
```

Y establecemos:

```
RUN_KAMAILIO=yes
```

Creamos el directorio para los archivos *pid*:

```
$ mkdir -p /var/run/kamailio
```

Por último, tendremos que crear el usuario para ejecutar Kamailio:

```
$ adduser --quiet --system --group --disabled-password \
      --shell /bin/false --gecos "Kamailio" \
      --home /var/run/kamailio kamailio

# set ownership to /var/run/kamailio
$ chown kamailio:kamailio /var/run/kamailio
```

Podremos iniciar, parar, comprobar el estado o reiniciar Kamailio:

```
$ /etc/init.d/kamailio start

$ /etc/init.d/kamailio stop

$ /etc/init.d/kamailio status

$ /etc/init.d/kamailio restart
```

ANEXO E. ENGLISH SUMMARY

1. INTRODUCTION

This project is carried out because of the increasingly use of VoIP (Voice over IP) communication. In Spain, according to CNMC (Comisión Nacional de los Mercados y la Competencia, National Commission for the Market and the Competition), there were 375,000 VoIP lines in 2007, and in 2016 just one ISP owned 2.9 million lines. We use these services every day, like Skype or WhatsApp calls, so it is essential to understand how they work.

Telephone companies, however, restrict its usage because this traffic is not transmitted through the PSTN (Public Switched Telephone Network). Through VoIP not only voice packets can be sent, but also video or data, and it does not need much infrastructure, it simply requires a stable internet connection. In a controlled context, such as a company, these advantages can be maximized in a way that conferences, calls or authorizations can be configured.

Given the importance of VoIP services, they are part of the elective course Audiovisual Services of the last year of the Telecommunication Technologies Degree. The purpose of this work is implementing a VoIP environment for university practices, according to RFC (Request For Comments) specifications for each protocol, that improves the existing one and which can be applied in the academic year 2018/19. It also serves to strengthen personal previous knowledge on the subject. These objectives are divided in sub-objectives:

- Setting a Domain Name System (DNS) to support users' and servers' identification through URI SIP.
- Support of multiple terminals, hardware and software. The university has VoIP Wi-Fi phones that permit assigning IP addresses, phone numbers or user names and passwords for identification. They implement the necessary protocols, such as SIP or RTP. There are also several applications for PC or smartphones that emulate them.
- Creation of conferences that support calling setup between multiple users.
- The environment is based on protocols and Internet standards. Specifically, SIP (Session Initiation Protocol) a signaling protocol that allows creating, modifying and terminating sessions between participants; SDP (Session Initiation Protocol) used to arrange a set of parameters that permits communication between them; RTP (Real Time Protocol) to transmit audio and video real-time.
- It is intended to be a low-cost system; therefore, its development is realized using open source tools and solutions.
- Validating the scenery developed through a set of appropriate tests that will be executed on the Telematic Engineering Department laboratory.

2. SYSTEM IMPLEMENTATION

The existing practices only consider the existence of terminals and a proxy/register server. This means some of the most important Internet elements, like DNS, are not considered. In the laboratory context described in this work exist, however, three fundamental elements: DNS server, proxy/register server and telephone center. In the design all these three components are set up into the same computer.

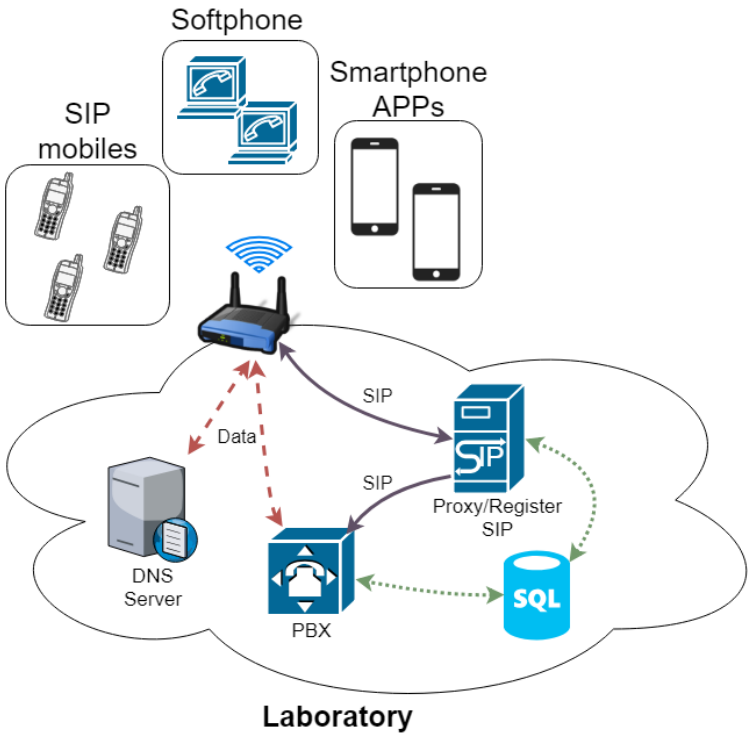

Fig. 43 System architecture

Mobile terminals can be connected through Wi-Fi to the laboratory network, in which the computers used already are.

The DNS server, executed using BIND, associates IP addresses of each device to a name in the domain in which they are, in this case “sip.uc3m.es”. All the SIP traffic goes through the proxy server, that is also the register server. The telephone center transmits data, maintains calls, conferences, etc. The proxy/server task is carried out by Kamailio, a software that, among other things, can manage multiple Asterisk instances, which is the one that acts as a telephone center. Both, Asterisk and Kamailio, access to the same SQL database, where users and their characteristics are stored.

2.1. BIND

BIND is an open source software that allows publishing and resolving DNS queries on the Internet. The first step is adding to a list of trusted devices the addresses that are going

to be used by clients, in this case from 172.1.2.1 to 172.1.2.20, and the servers 172.1.2.99 and 172.1.2.100. The second server is optional but recommended in terms of security.

Then, the forward and reverse zones are added, this is associating “sip.uc3.es” to those addresses. The main resources used are A, SRV, NS and PTR, which relate those directions.

2.2. Asterisk

Asterisk is an open source framework that acts as a PBX (Private Brach Exchange) and used to create VoIP servers.

Four files must be configured to achieve the desired performance:

- sip.conf: contains general parameters about the SIP context. Addresses, ports, call plans characteristics or protocols can be specified. Set here:

```
rtcachefriends=yes
udpbindaddr=172.1.2.99:5080
tcpbindaddr=172.1.2.99:5080
laguage=es
```

- asterisk.conf: if the language has been changed in the previous case, here should be changed too:

```
defaultlanguage = es
```

- extensions.conf: in this file the call plans are created so that users can call one another or create conferences. As each user has a number between 1 and 20 assigned in DNS, the phone numbers will be 00XX, where XX is one of them. To make conferences users must dial a number between 8000 and 8999.

- confbridge.conf: conference and user features are described here. For example, users can be admins, mute their microphone when talking or change the volume.

2.3. Kamailio

Kamailio is an open source SIP server that permits the management of thousands of calls each second. It is integrated with Asterisk with the intention of facilitate scalability and performance. Kamailio acts as proxy server and Asterisk handles calls, users, conferences, etc.

2.4. User creation

To add new users able to use this service they need to be added to a database used by Asterisk, in which the username, password, address, and context must be described. Kamailio uses this database too in order to make the authentication process.

2.5. Phones

Phone configuration can be easily done in a computer through an interface, which is accessed typing the address of the phone in the internet browser. In each of them the address desired, DNS server IP, port, name, password or proxy domain names can be selected.

3. TESTS

Once everything is configured correctly we can verify that it works as expected, fulfilling the current standards defined in RFCs. Some specific tests must be applied to each one of the previous sections:

- Identification of users and VoIP servers through URI SIP.
- Terminals registration on the server.
- Calls between two terminals and verification of possible answers.
- Conference between two or more users.

The first step is initializing Kamailio and Asterisk:

```
$ /etc/init.d/kamailio start

$ /etc/init.d/asterisk start
```

We can use the console Asterisk provides to check information related to the state of users, calls, conferences, etc:

```
$ asterisk -vr
```

3.1. DNS

Phones register using the domain name “sip.uc3m.es”, so when they are switched on they send a DNS request to the configured address where the server is, which is at “172.1.2.99”, that also corresponds to the proxy server.

It is a standard request, full and with recursion; the query name is “sip.uc3m.es” and the query type is “A”, so it must return the IP. As it is a question, the other fields in the header are empty. In addition to the name solicitated, it returns list of authoritative servers as well.

3.2. Register

To get registered, a set of SIP packets is exchanged between UA and the proxy server when the terminal is started. The client must send the data to the proxy server, which runs Kamailio and listens in the direction “172.1.2.99” and port “5060”. The first message sent is answered with a “401 Unauthorized”. This means that the server has checked the

database and asks the UA to send information necessary for the authentication, which are the corresponding username and password.

As this is the first message sent, the header fields “Via”, “From” and “To” contain all the same information, a URI SIP with the address of the emitter. The “From” field also contains the tag which is associated to it and identifies it in the session. In “Contact” is specified that the requests must be sent to the same direction and the expire time is 120 seconds. Finally, as it was mentioned before, the information to do the registration is in the “Authorization” header.

In this case, the answer from the server is “200 OK”. The header fields are the same as the request, except for the “To”, where the tag of the server is included. It is also indicated that this server is Kamailio.

To unregister the process is very similar. The UA sends the same information as it did before, but now sets the expire time to 0 seconds.

3.3. Call

Calls between users begin with a SIP INVITE request, in which the emitter specifies the destination and conditions of them using SDP. As before, the proxy answers to the first message indicating that authentication is required. When username and password are confirmed, the server admits the invitation and routes it to the destination.

The message sent from the caller contains the destination URI in the “To” field, for example “sip:0008@172.1.2.8”. The other components of the header have the same information as in the register. Regarding the message body, it is composed of SDP protocol, where the session creator is indicated, and several choices for audio codecs are presented, such as G729, PCMU or PCMA. The messages are transmitted from the server to the receiver as they are, making changes to add its own information only in the “Via” field.

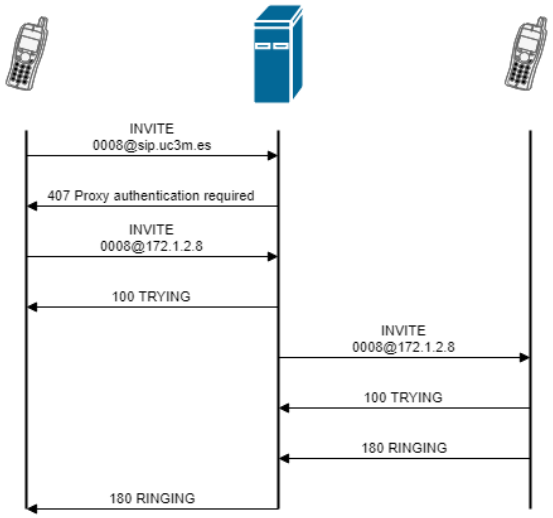

Fig. 44 Call setup

At this point, three things can happen:

3.3.1. OK

The receiver accepts the call with some of the conditions available. The termination requires to be done using SIP BYE messages. When it is accepted, the message body contains a single “a” field, therefore, communication will be carried out with the codec indicated. The OK answer is propagated backwards from the receiver and, once it gets to the emitter, it sends an ACK. Communication finishes if one of the components hangs up, by sending BYE, and is confirmed to be done when the other component has received that solicitude.

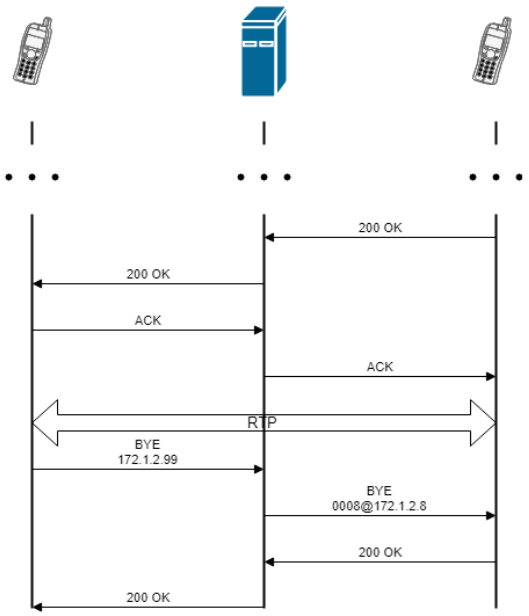

Fig. 45 Accepted call

Taking one of the RTP messages exchanged it is possible to check that, effectively, the codec used in the payload is the one agreed on previously during the session establishing, where it was selected G729; contains a sequence number and an identifier of the source that permits the rearrangement in case there is any error. RTCP packets type are Sender Reports, so they send information about what is sent and received.

3.3.2. BUSY

The receiver rejects the call by sending a “486 BUSY” to which the server answers with an ACK, and resends it to the emitter, who will answer with another ACK.

3.3.3. CANCEL

In this case, it is the emitter who hangs up before the receiver can answer. The server sends a *2xx* type message that indicates the call has been successfully cancelled. Then, it resends the CANCEL request to the receiver, who does the same, signaling that it has been received. After that, it stops ringing and goes back to the initial state, which is announced by a message “487 REQUEST TERMINATED”, propagated backwards.

3.4. Conference

Regarding conferences between multiple users, they are the ones who must generate them. This can be done by dialing the identifier of the conference that is going to be created. It is configured to do so in Asterisk if the number dialed is between 8000 and 8999. When this is executed, the terminal sends the server a SIP INVITE with that URI, for example “sip:8762@sip.uc3m.es”, which is received by Kamailio and processed by Asterisk as if it were a normal call.

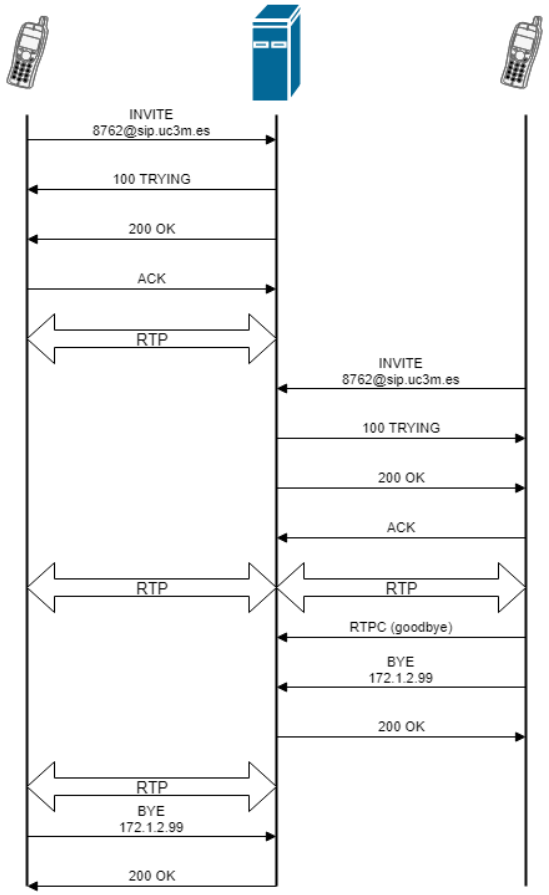

Fig. 46 Diagram of a conference

The creator is added automatically to that conference and it starts transmitting RTP messages to the server. Any other user can join the conference since the moment it was created dialing the same identifier through an INVITE. In this case, there will be no ringing, instead, the call is directly accepted after a “trying” message and “OK”. Unlike normal calls, conditions are agreed upon server and each participant every time a new one calls. Conferences, in contrast to normal calls, use codec G.711 which has less compression.

4. CONCLUSSION

This project has consisted in the creation of a VoIP communication environment for teaching practices that allows, together with DNS service, voice communication between terminals based on SIP. At the beginning two objectives were contemplated, on the one hand, dig into VoIP communication knowledge in realistic circumstances; on the other, being capable of using it in practices.

A server that permits different users to interconnect and communicate has been stablished, whose control and description depends entirely on who installs it, so once it has been used in the course, students will be capable of checking the working mode of calls, conferences and DNS applied to VoIP, since from this side, it is possible to access to all the information that is exchanged, like processed calls, active users, etc. This might help understand the standards and theory explained in class.

The client, on its side, can perform any of the characteristic functions in a telephone communication context, such as registration or call setup. This can be done with no need of the inner working system, just one IP address or domain name and a port number.

Personally, this project has helped to consolidate previous knowledge about DNS systems and SIP related protocols implicated in VoIP. Also, as the project was progressing, new challenges, that were not considered at the beginning, have been appearing and, whose resolution has helped to learning new concepts and consider diverse point of views for the situations.

Once the project is finished it is possible to say that all the requirements determined at the beginning have been fulfilled. This permits that in the future new activities based on what have been explained before. For these reasons, it is possible to say that the objectives have been achieved.

4.1. Future

From this point new upgrades can be added to develop new functionalities in the already established system. Among these options:

- Voicemail: one of the tables present in SQL is voicemail. It is possible to configure a context in which, in case a call is not established after an INVITE request, the initiator can record an audio message for the receiver, who will be capable of accessing to it dialing a certain number.
- Videocall: some of the options when negotiating a call include exchange of real-time video. However, its use has not been possible in this work because of the limitations on the devices used.
- Raspberry Pi: there exists the possibility of installing this service in some device of this type so that its connection and displacement is easier, instead of initiating everything independently every time the system is shut down. This way, the setup can work just by switching up that device.

- Charge distribution: one of the advantages of Kamailio is the choice to make a balanced distribution the work between several Asterisk instances. This allows the number of requests to grow considerably and so this can be widely used, something that can be interesting to explore in a practice.
- DHCP: Dynamic Host Configuration Protocol is a protocol that permits assigning the clients a set of available IP addresses automatically. Because of the limitations, this could not be applied in the practice either because the router used was configured so that IP assignation is fixed.